SIEMENS

SINUMERIK

SINUMERIK 840D sl / 828D Extended Functions

Function Manual

Valid for

Controls
SINUMERIK 840D sl / 840DE sl
SINUMERIK 828D

Software	version
CNC software	4.5 SP2

03/2013
6FC5397-1BP40-3BA1

Legal information

Warning notice system

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

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

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

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

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-1BP40-3BA1
Ⓟ 05/2013 Technical data subject to change

Preface

SINUMERIK documentation

The SINUMERIK documentation is organized in the following categories:

- General documentation
- User documentation
- Manufacturer/service documentation

Additional information

You can find information on the following topics at www.siemens.com/motioncontrol/docu:

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

Please send any questions about the technical documentation (e.g. suggestions for improvement, corrections) to the following address:

docu.motioncontrol@siemens.com

My Documentation Manager (MDM)

Under the following link you will find information to individually compile OEM-specific machine documentation based on the Siemens content:

www.siemens.com/mdm

Training

For information about the range of training courses, refer under:

- www.siemens.com/sitrain

  SITRAIN - Siemens training for products, systems and solutions in automation technology
- www.siemens.com/sinutrain

  SinuTrain - training software for SINUMERIK

FAQs

You can find Frequently Asked Questions in the Service&Support pages under Product Support. http://support.automation.siemens.com

SINUMERIK

You can find information on SINUMERIK under the following link:

www.siemens.com/sinumerik

Target group

This publication is intended for:

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

Benefits

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

Standard version

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

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

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

Technical Support

You will find telephone numbers for other countries for technical support in the Internet under http://www.siemens.com/automation/service&support

Information on structure and contents

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
  - Documentation overview
- Index of terms

Note

For detailed descriptions of data and alarms see:

- For machine and setting data:
  Detailed description of machine data (only electronically on DOConCD or DOConWEB)
- For 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)
- For alarms:
  Diagnostics Manual

Notation of system data

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

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

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

Quantity structure

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

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

Data types

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

Type	Meaning	Range of values
INT	Signed integers	-2.147.483.648 ... +2.147.483.647
REAL	Numbers with decimal point	$\approx \pm 5{,}0 \cdot 10^{-324} \ldots \approx \pm 1{,}7 \cdot 10^{+308}$
BOOL	Boolean values	TRUE (≠0) , FALSE (0)
CHAR	ASCII characters and bytes	0 ... 255 or -128 ... 127
STRING	Character string, null-terminated	Maximum of 400 characters + /0 (no special characters)
AXIS	Axis names	All axis names available in the control system
FRAME	Geometrical parameters for moving, rotating, scaling, and mirroring	---
Note Arrays can only be formed from similar elementary data types. Up to 3-dimensional arrays are possible.		

Contents

A4: Digital and analog NCK I/Os for SINUMERIK 840D sl

1

1.1 Introduction

Function

Signals can be read and output in the interpolation cycle via the inputs/outputs of the fast digital and analog NCK I/Os. The following functions can be executed with these signals, for example:

- Several feedrate values in one block
- Several auxiliary functions in one block
- Rapid retraction on final dimension
- Axis-specific delete distance-to-go
- Program branches
- Rapid NC Start
- Analog calipers
- Position switching signals
- Punching/nibbling functions
- Analog-value control

Hardware

The SINUMERIK 840D sl hardware has three 14-pin I/O interfaces X122, X132 and X142 on the NCU on-board. Only X142 is available as a fast NCK I/O.

There are four fast digital inputs and four outputs on the X142 that can be addressed via the first address byte and via the $A_IN[1...4] and $A_OUT[1...4] system variables.

ET 200 modules can also be connected to the PROFIBUS DP/MPI interfaces X126 and X136. This enables the number of digital and analog NCK I/Os to be expanded by 32 and 8 respectively. These NCK I/Os are called external NCK I/Os in the following.

Table 1- 1 Maximum number of digital and analog NCK I/Os

	Total	NCU on-board	External NCK I/Os
Digital inputs	36	4	32
Digital outputs	36	4	32
Analog inputs	8	-	8
Analog outputs	8	-	8

References:
For further information about the hardware specification, refer to:

- SINUMERIK 840D sl NCU Manual
- SIMATIC ET 200S FC Operating Instructions

Access to NCK I/Os:

The fast digital and analog NCK I/Os can be accessed in the following ways:

- Access via PLC (Page 27) (for on-board I/Os and PROFIBUS I/Os)
- Access via PROFIBUS (Page 54) (for PROFIBUS I/Os)

Monitoring

The following monitoring functions are active for the external NCK I/Os:

- During run-up:
  – Check whether the arrangement of components of the I/O modules in the ET 200 modules matches the MD assignments.
- During cyclic operation:
  – Sign-of-life monitoring in interpolation cycles
  – Module monitoring in interpolation cycles
  – Temperature monitoring

In the event of a fault, NC-Ready is canceled and an alarm is output.

Response to faults

The digital and analog NCK outputs are switched to a safe status (0 V at output) in the event of faults (e.g. NC-Ready = 0) in the NCU or power failures.

1.2 Access via PLC

1.2.1 Brief description

Configuring NCK I/Os

During commissioning, the number and the hardware assignment of the addressable digital and analog NCK I/Os and their assignment to NC functions is defined via machine data.

For further information, see "Configuration of the NCK I/Os (Page 28)".

Data exchange

The signals and analog values of the digital and analog I/Os are available in the NC (part program, ASUB, synchronized action, etc.) in the form of system variables.

- $A_IN [<n>] Read digital NCK input <n> (where <n> = 1...4 and 9...40)
  For further information, see "NCK digital inputs (Page 31)".
- $A_OUT [<n>] Read/write digital NCK output <n> (where <n> = 1...4 and 9...40)
  For further information, see "NCK digital outputs (Page 32)".
- $A_INA [<n>] Read analog NCK input <n> read (where <n> = 1...8)
  For further information, see "NCK analog inputs (Page 37)".
- $A_OUTA [<n>] Read/write analog NCK output <n> read (where <n> = 1...8)
  For further information, see "NCK analog outputs (Page 39)".

<n> = number of the NCK input/output

Note

When these system variables are read by the part program, a preprocessing stop (STOPRE) is initiated inside the control.

Comparator inputs

In addition to the digital and analog NCK inputs, 16 internal comparator inputs (comparator input bytes 1 and 2) are also available.

The signal state of a comparator input is formed by comparing an analog input signal with a threshold value within a setting data.

For further information, see "Comparator inputs (Page 45)".

PLC I/Os for direct addressing by NCK

Up to 32 bytes for digital input signals and analog input values, and up to 32 bytes for digital output signals and analog output values can be addressed directly by the part program. These bytes must be taken into account when the PLC is configured. They are processed directly by the PLC operating system. As a result, the time taken to transfer signals between the NC and PLC I/O modules is of an order of magnitude of 0.5 ms.

Note

The output bytes specified for the NCK may not be write-accessed by the PLC user program, as the access operations between the NCK and PLC would be uncoordinated.

For further information, see "Direct PLC I/Os, addressable from the NC (Page 48)".

1.2.2 Configuration of the NCK I/Os

Number of active NCK I/Os

The number of addressable digital NCK I/O bytes and analog I/Os is defined with the machine data:

• MD10350 $MN_FASTIO_DIG_NUM_INPUTS	Number of active digital NCK input **bytes** (max. 5)
• MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS	Number of active digital NCK output **bytes** (max. 5)
• MD10300 $MN_FASTIO_ANA_NUM_INPUTS	Number of active analog NCK inputs (max. 8)
• MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS	Number of active analog NCK outputs (max. 8)

Corresponding alarms are generated if the part program addresses I/Os that have not been defined in the above machine data.

The NCK I/Os do not have to actually exist in the hardware. In this case the signal states or the binary analog values are set to "zero" in a defined way inside the NCK. However, these values can be changed by the PLC.

Hardware assignment of the external NCK I/Os

The assignment of the I/O signal modules or I/O modules to the external NCK I/Os is performed via the machine data:

- MD10366 $MN_HW_ASSIGN_DIG_FASTIN[<n>] — HW assignment for external digital inputs
- MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[<n>] — HW assignment for external digital outputs
- MD10362 $MN_HW_ASSIGN_ANA_FASTIN[<n>] — HW assignment for external analog inputs
- MD10364 $MN_HW_ASSIGN_ANA_FASTOUT[<n>] — HW assignment for external analog outputs

<n> = Index for addressing the external digital I/O bytes (0 ... 3) or the external analog I/Os (0 ... 7)

Example:

Two additional input bytes and one additional output byte are configured in the NC for the data exchange with the digital I/Os of an external PROFIBUS module.

Hardware assignment:

MD10366 $MN_HW_ASSIGN_DIG_FASTIN[0]='H5000200' ; for $A_IN[9] ... [16]

MD10366 $MN_HW_ASSIGN_DIG_FASTIN[1]='H5000201' ; for $A_IN[17] ... [20]

MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[0]='H5000200' ; for $A_OUT[9] ... [16]

'H5000000' specifies that the input/output is on the PROFIBUS.

The lower-order digits specify the start address of the assigned module as a logical PROFIBUS address in hexadecimal format.

For the configuration in the SIMATIC Manager, the same logical address is assigned in decimal format (e.g. the value 'H202' corresponds to the logical address '514').

Number of active NCK I/O bytes:

MD10350 $MN_FASTIO_DIG_NUM_INPUTS = 3

MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS = 2

Note

The on-board byte must also be counted for the configuration of MD10350 or MD10360!

Weighting factor for the analog NCK I/Os

The weighting factor can be used to adapt each individual NCK I/O to the AD or DA converter of the analog I/O module used:

- MD10320 $MN_FASTIO_ANA_INPUT_WEIGHT[<n>] — Weighting factor for the analog NCK inputs

  For further information, see "NCK analog inputs (Page 37)".
- MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[<n>] — Weighting factor for the analog NCK outputs

  For further information, see "NCK analog outputs (Page 39)".

Assignment to NC functions

Several NC functions are dependent on the functionality of the NCK I/Os.

The assignment of the NCK I/Os used for these NC functions is performed function-specifically via machine data, e.g. for the "Multiple feedrates in one block" function via the machine data:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN

A byte address should be specified in the machine data for digital inputs/outputs. The assignment is always byte-by-byte.

Byte address	Assignment for the digital NCK I/Os			
0	None			
1	1 to 4 (on-board I/O)	and	5 to 8	(NCK output without hardware)
2	9	to	16	(external NCK I/Os)
3	17	to	24	(external NCK I/Os)
4	25	to	32	(external NCK I/Os)
5	33	to	40	(external NCK I/Os)
128	Inputs 1 to 8 of comparator byte 1			
129	Inputs 9 to 16 of comparator byte 2			

Note

Multiple assignments

Multiple assignments of inputs are not monitored.

Multiple assignments of outputs are checked during run-up and indicated by an alarm.

1.2.3 NCK digital inputs/outputs

1.2.3.1 NCK digital inputs

Function

The workpiece-machining program sequence can be controlled by external signals via digital NCK inputs.

The signal state of digital input <n> can be scanned directly in the part program using system variable $A_IN [<n>].

The signal state at the hardware input can be changed from the PLC user program.

Applications

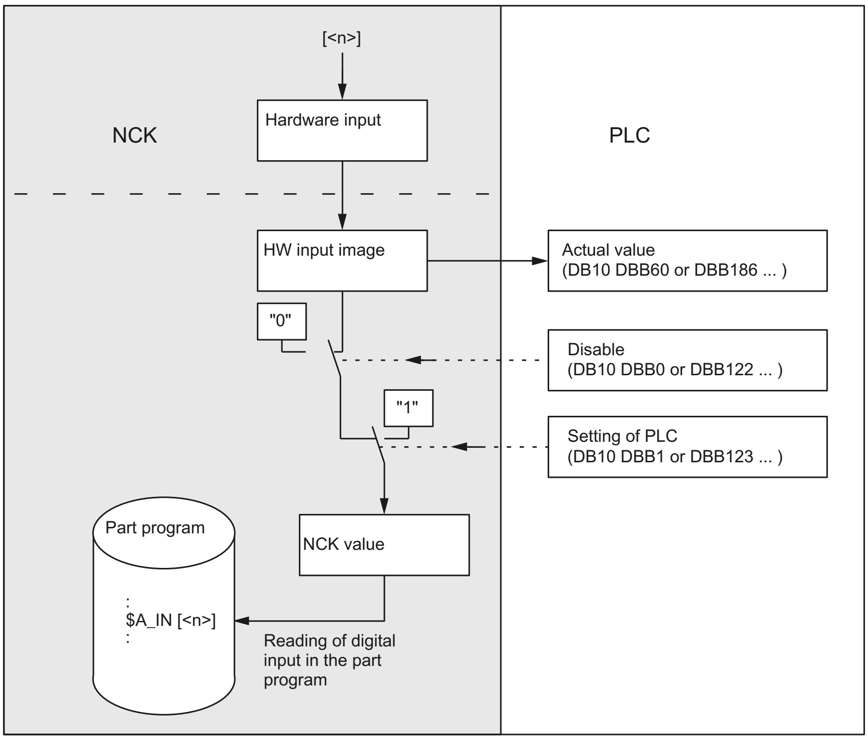
Digital NCK inputs are used, for example, for the following NC functions:

- Delete distance-to-go with positioning axes
- Fast program branching at the end of block
- Programmed read-in disable
- Several feedrates in one block

References:
Function Manual, Synchronized Actions

Signal flow

The following figure illustrates the signal flow for the digital NCK inputs.



Read actual value

The signal state of the digital NCK inputs is sent to the PLC:

DB10, DBB60 or DBB186 ... (actual value for digital NCK inputs)

The actual value reflects the actual state of the signal at the hardware input. The influence of the PLC is ignored for the "actual value".

Disable input

Digital NCK inputs can be disabled individually from the PLC user program:

DB10 DBB0 or DBB122 ... (block the digital NCK inputs)

In this case, they are set to a defined "0" inside the control.

Set input from PLC

The PLC can also set each digital input to a defined "1" signal:

DB10 DBB1 or DBB123 ... (setting of the digital NCK inputs from the PLC)

As soon as this interface signal is set to "1", the signal state at the hardware input or the input disable is inactive.

Behavior during POWER ON / reset

After POWER ON and reset, the signal level at the respective input is passed on. If necessary, the PLC user program can disable or set the individual inputs to a defined "1" as described above.

1.2.3.2 NCK digital outputs

Function

Time-critical switching operations can be triggered very quickly via the digital NCK outputs, bypassing the PLC cycle times for the relevant machining and program-controlled (e.g. with the block change).

The signal state of digital output <n> can be set or read again directly in the part program using system variable $A_OUT[<n>].

There are also several ways of changing this set signal state via the PLC.

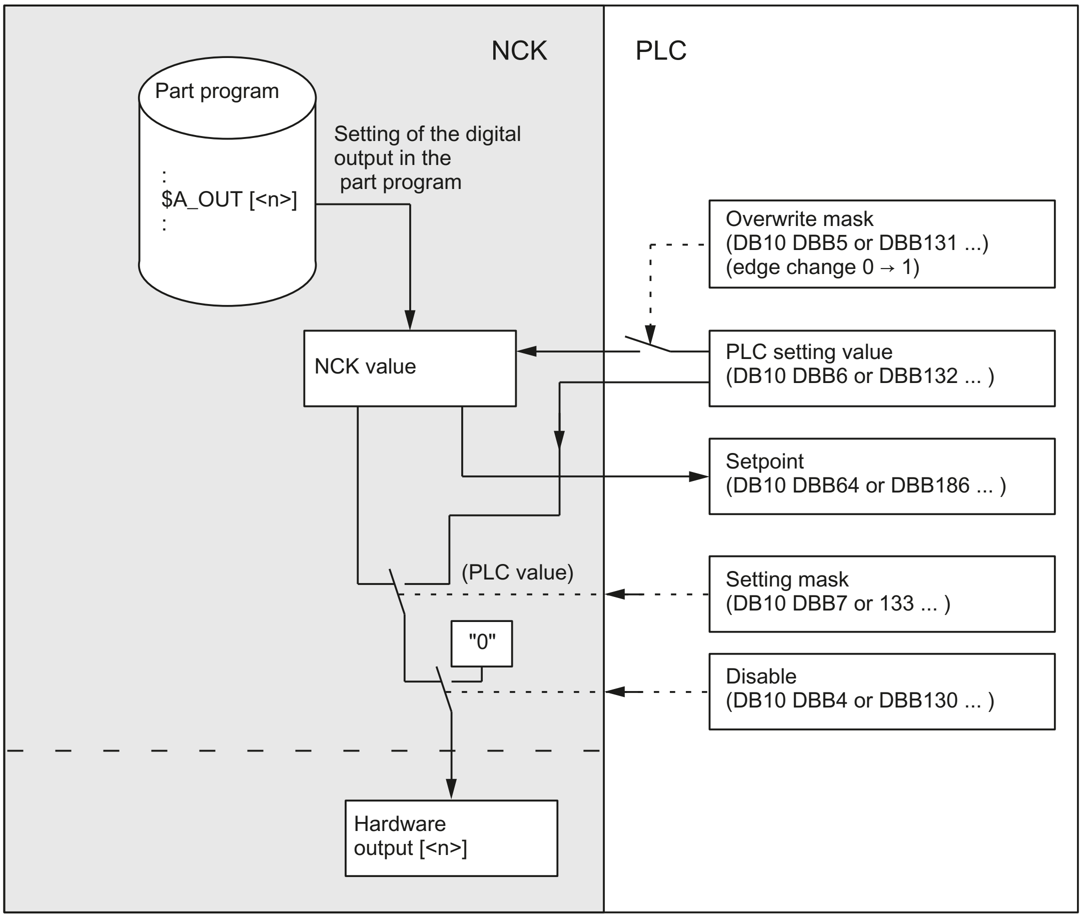
Applications

Digital NCK outputs are used, for example, for the following NC functions:

- Position switching signals (see Section "N3: Software cams, position switching cycles - only 840D sl (Page 535)")
- Output of comparator signals

Signal flow

The following figure illustrates the signal flow for the digital NCK outputs.



Overwrite mask

Every output that can be set by the NC part program can be overwritten from the PLC using the overwrite mask. The previous "NCK value" is lost.

The following sequence has to be carried out to overwrite the NCK value from the PLC:

1. The relevant PLC interface output has to be set to the required signal status.

   DB10 DBB6 or DBB132 ... (setting value of the digital NCK outputs from the PLC)

2. The "setting value" becomes the new "NCK value" when the overwrite mask for the relevant output is activated (edge change 0 → 1).

   DB10 DBB5 or DBB131 ...

   This value remains operative until a new NCK value is programmed (from the PLC or from the part program).

Setting mask

A PLC setting for each output can determine whether the current "NCK value" (e.g. as specified by the NC part program) or the "PLC value" specified via the setting mask should be output at the hardware output.

The following sequence has to be carried out to define the "PLC value":

1. The relevant PLC interface output has to be set to the required signal status.

   DB10 DBB6 or DBB132 ... (setting value of the digital NCK outputs from the PLC)

2. The setting mask must be set to "1" for the relevant output:

   DB10 DBB7 or DBB133 ... (setting mask of the digital NCK outputs)

Unlike the overwrite mask, the NCK value is not lost when a value is set in the setting mask. As soon as the PLC sets "0" in the corresponding setting mask, the NCK value becomes active again.

Note

The same setting value is used at the PLC interface for the overwrite and setting masks. Therefore, an identical output signal state is the result if the signal state is changed simultaneously in the overwrite and setting masks.

Disable output

Digital NCK outputs can be disabled individually from the PLC user program:

DB10 DBB4 or DBB130 ... (block the digital NCK outputs)

In this case, the "0" signal is output at the hardware output.

Read setpoint

The current "NCK value" of the digital outputs can be read by the PLC user program:

DB10, DBB64 or DBB186 ... (setpoint for digital NCK outputs)

Please note that this setpoint ignores disabling and the PLC setting mask. Therefore, the setpoint can differ from the actual signal state at the hardware output.

Behavior at program end / reset

At the end of the program or on reset, a specific default value can be assigned by the PLC user program to every digital output in accordance with requirements, using the overwrite mask, setting mask or disable signal.

Response to POWER ON

After POWER ON, the digital outputs are set to "0" in a defined manner. This can be overwritten from the PLC user program according to the specific application using the overwrite or setting mask.

Digital NCK outputs without hardware

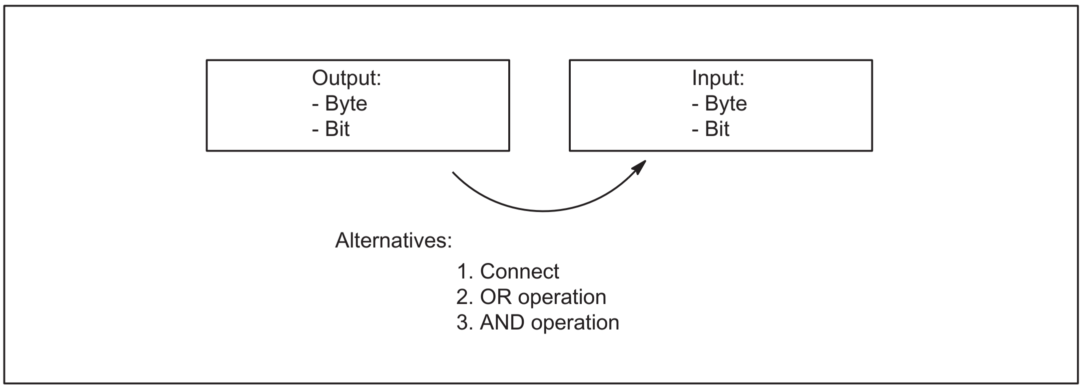
If digital NCK outputs, as defined via MD10360, are written by the part program, but are not available as hardware, no alarm is output. The NCK value can be read by the PLC (DB10 DBB64 or DBB186 ...).

1.2.3.3 Connection and logic operations of fast digital NCK I/Os

Function

Fast NCK I/O inputs can be set by software as a function of fast-output signal states.

Overview:



Connecting

The NCK I/O fast input is set to the signal state of the assigned fast output.

OR operation

The NCK I/O fast input adopts the signal state as a result of the ORing of the output signal with the assigned input signal.

AND operation

The NCK I/O fast input adopts the signal state as a result of the ANDing of the output signal with the assigned input signal.

Special cases

- If several output bits are assigned to the same input bit, then the one with the highest MD index becomes effective.
- If inputs or outputs are specified which do not exist or are not activated, then the assignment is ignored without an alarm. Checking of the active bytes of the NCK I/Os is performed via the entries in machine data:

  MD10350 $MN_FASTIO_DIG_NUM_INPUTS

  MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

Defining assignments

The assignments are specified via machine data:
MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT[n]

n: can accept values 0 to 9, so up to **10** assignments can be specified.

Two hexadecimal characters are provided for specifying the byte and bit of an output and an input.

Specifying 0, A and B in input bits 12 - 15 results in the following **logic operations**:

0	Connecting
A	AND operation
B	OR operation

FASTIO_DIG_SHORT_CIRCUIT[n]

	Output		Input	
	Bit	Byte	↑ Type of logic op | Bit	Byte
Bit	24-31	16-23	8-15	0-7

Examples

Connect:

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '04010302H'

Output 4, byte 1, connect to

Input 3, byte 2

AND operation:

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '0705A201H'

Output 7, byte 5 AND operation with

Input 2, byte 1

OR operation:

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '0103B502H'

Output 1, byte 3 OR operation with

Input 5, byte 2

1.2.4 NCK analog I/Os

1.2.4.1 NCK analog inputs

Function

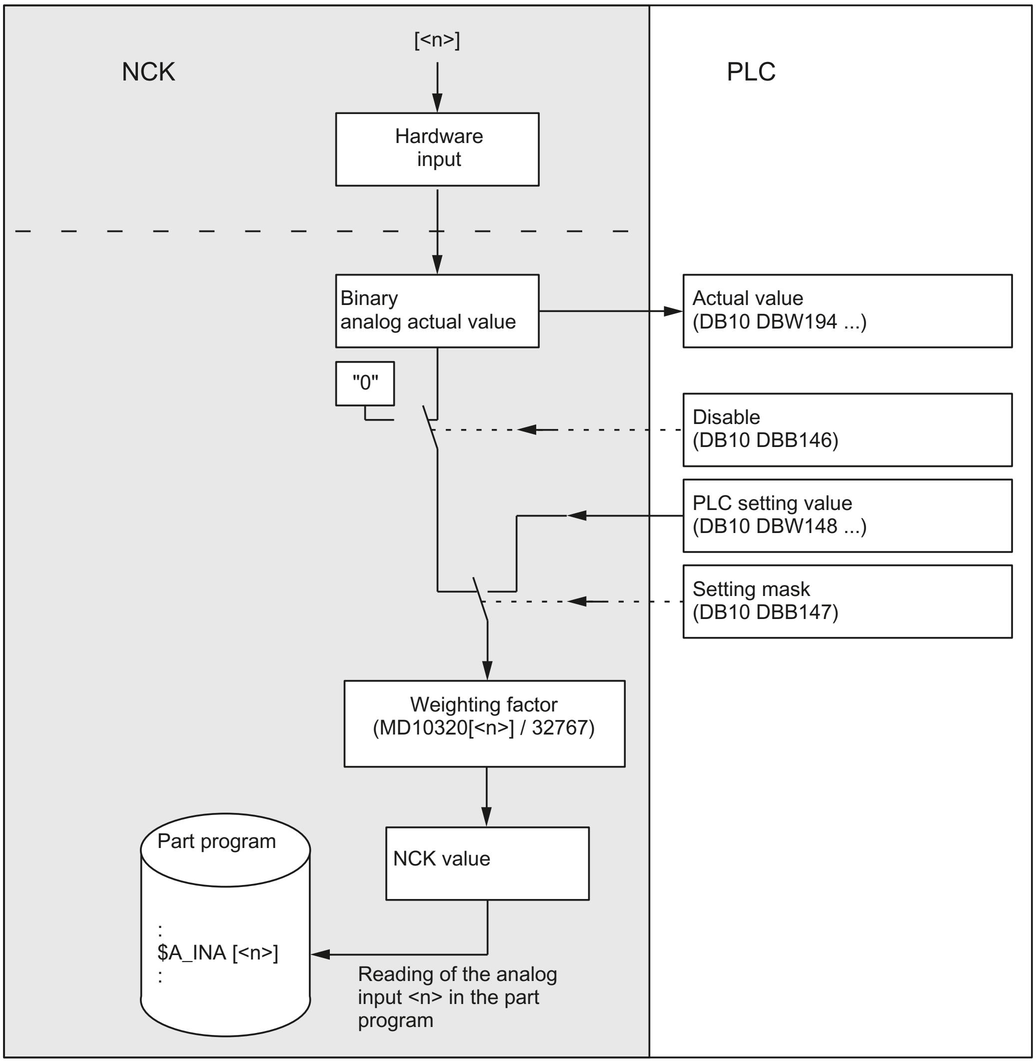
The value of the analog NCK input [<n>] can be accessed directly in the part program using system variable $A_INA[<n>].

The analog value at the hardware input can be controlled from the PLC user program.

Applications

The analog NCK inputs are used particularly for grinding and laser machines, e.g. for the "analog calipers" NC function.

Signal flow



Read actual value

The analog values that are actually present at the hardware inputs are are sent to the PLC:

DB10 DBW194 ... 208 (actual value of the NCK analog input)

The possible influence of the PLC is ignored for the "actual value".

Disable input

Analog NCK inputs can be disabled individually from the PLC user program:

DB10 DBB146 (Disable analog NCK inputs)

In this case, they are set to a defined "0" inside the control.

Set input from PLC

The PLC can also specify a value for each analog NCK input:

DB10 DBB147 (analog value specification for NCK from the PLC)

As soon as this interface signal is set to "1", the setting value set by the PLC becomes active for the corresponding analog input.

DB10 DBW148-162 (setpoint from the PLC for analog NCK input)

The analog value at the hardware input or the input disable is then inactive.

Weighting factor

The weighting factor can be used to adapt each individual NCK input to the various AD converters (depending on the I/O module) for reading in the part program:

MD10320 $MN_FASTIO_ANA_INPUT_WEIGHT[<n>]

In this machine data, it is necessary to enter the value x that is to be read in the part program with the system variable $A_INA[<n>], if the corresponding analog input <n> is set to the maximum value or if the value 32767 is set for this input via the PLC interface. The voltage level at the analog input is then read with system variable $A_INA[<n>] as a numerical value with the unit millivolts.

Note

Application for analog NCK inputs without hardware:

With a weighting factor of 32767, the digitized analog values for part program and PLC access are identical. In this way, it is possible to use the associated input word for 1:1 communication between the part program and the PLC.

Binary analog-value display

See "Representation of the NCK analog input/output values (Page 43)".

Behavior during POWER ON / reset

After POWER ON and reset, the analog value at the respective input is passed on. If necessary, the PLC user program can disable or set the individual inputs to a setpoint.

Analog NCK input without hardware

The following value is read in the case of part program access to analog NCK inputs that are defined via MD10300, but are not available as hardware inputs:

- The setpoint specified from the PLC (if the IS "Analog value input for NCK from the PLC" is set to "1")
- 0 volts (if the IS "Analog value input for NCK from the PLC" is not set)

This makes it possible to use the functionality of the analog NCK inputs from the PLC user program without I/O hardware.

1.2.4.2 NCK analog outputs

Function

Analog values to be output very quickly via the analog NCK outputs, bypassing the PLC cycle times.

The value of the analog output <n> can be defined directly in the part program using system variable $A_OUTA[<n>].

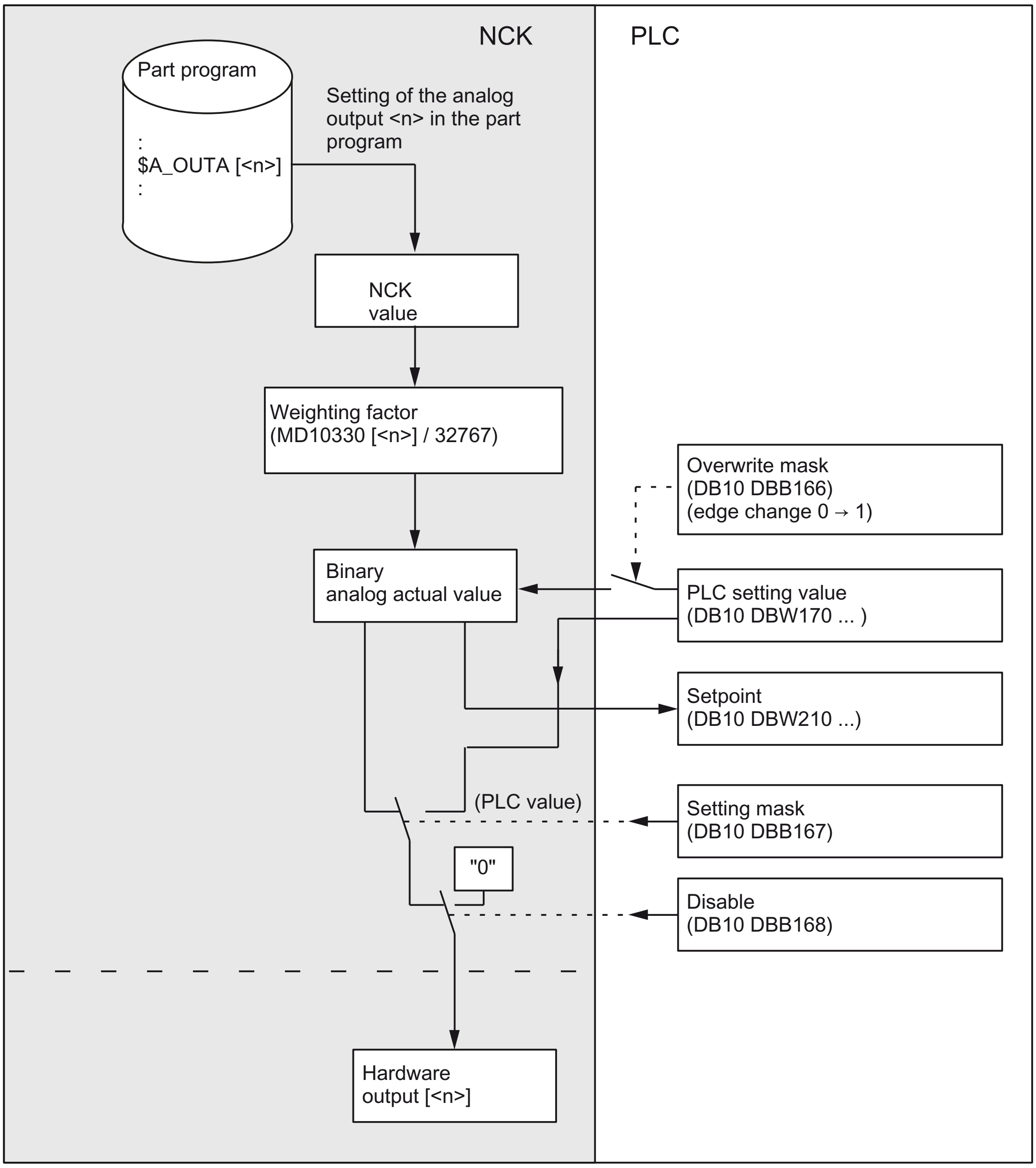
Before output to the hardware I/Os, the analog value set by the NCK can be changed by the PLC.

Application

The analog NCK outputs are used in particular for grinding and laser machines.

Signal flow

The following figure illustrates the signal flow for the analog NCK outputs.



Overwrite mask

Every NCK analog value set by the part program can be overwritten from the PLC using the overwrite mask. The previous "NCK value" is lost.

The following sequence has to be carried out to overwrite the NCK value from the PLC:

1. The relevant PLC interface output <n> has to be set to the required analog value.

   DB10 DBW170 … (setpoint from the PLC for analog output <n> of the NCK)

2. The "setpoint from the PLC" becomes the new "NCK value" when the overwrite mask for the relevant analog output is activated (edge change 0 → 1).

   DB10 DBB166 (overwrite mask of analog NCK outputs)

This value remains operative until a new NCK value is programmed (from the PLC or from the part program).

Setting mask

A PLC setting for each output can determine whether the current "NCK value" (e.g. as specified by the NC part program) or the "PLC value" specified via the setting mask should be output at the hardware analog output.

The following sequence has to be carried out to define the "PLC value":

1. The relevant PLC interface output has to be set to the required analog value.

   DB10 DBW170 … (setpoint from the PLC for analog output <n> of the NCK)

2. The setting mask must be set to "1" for the relevant analog output:

   DB10 DBB167 (Setting screen form of analog NCK outputs)

Unlike the overwrite mask, the NCK value is not lost when a value is set in the setting mask. As soon as the PLC sets "0" in the corresponding setting mask, the NCK value becomes active again.

Note

The same setpoint is used at the PLC interface for the overwrite and setting masks.

Disable output

Analog NCK outputs can be disabled individually from the PLC user program:

DB10 DBB168 (disable analog NCK outputs)

In this case, the "0" signal is output at the hardware output.

In this case, **0 volt** is output at the analog output.

Read setpoint

The current "NCK value" of the analog outputs can be read from the PLC user program:

DB10 DBW210 … (setpoint of the analog output <n> of the NCK)

Please note that this setpoint ignores disabling and the PLC setting mask. Therefore, the setpoint can differ from the actual analog value at the hardware output.

Weighting factor

The weighting factor can be used to adapt each individual NCK output to the various DA converters (depending on the I/O module) for programming in the part program:

MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[<n>]

In this machine data, it is necessary to enter the value x that is to cause the analog output <n> to be set to the maximum value or the value 32767 to be set for this output in the PLC interface, if $A_OUTA[n] = x is programmed. The value set with system variable $A_OUTA[<n>] then generates the corresponding voltage value at the analog output in millivolts.

Example:

Analog-value range is 10 V (maximum modulation);

MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[<n>] = 10000 (default value)

$A_OUTA[1] = 9500; 9.5 V is output at analog NCK output 1

$A_OUTA[3] = -4120; -4.12 V is output at analog NCK output 3

Note

Application for analog NCK outputs without hardware:

With a weighting factor of 32767, the digitized analog values for part program and PLC access are identical. In this way, it is possible to use the associated output word for 1:1 communication between the part program and the PLC.

Binary analog-value display

See "Representation of the NCK analog input/output values (Page 43)".

Behavior at program end / reset

At the end of the program or on reset, a specific default value can be assigned by the PLC user program to every analog output in accordance with requirements, using the overwrite mask, setting mask or disable signal.

Response to POWER ON

After POWER ON, the analog outputs are set to "0" in a defined manner. This can be overwritten from the PLC user program according to the specific application using the overwrite or setting mask.

Analog NCK outputs without hardware

If analog NCK outputs, as defined via MD10310, are written by the part program, but are not available as hardware, no alarm is output. The NCK value can be read by the PLC (DB10 DBB210 …).

1.2.4.3 Representation of the NCK analog input/output values

The digitized analog values are represented at the NC/PLC interface as fixed-point numbers (16 bits including sign) in the two's complement.

Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Significance	SG	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

SG: Sign

Minimum value	Maximum value
-32768_D	32767_D
8000_H	$7FFF_H$

Increment

For a resolution of 16 bits and a nominal range of ±10 V, the increment is:

$20 V / 2^{16} = 20 V / 65536 \approx 0.305 mV$

Resolutions < 16 bits

If the resolution of an analog module is less than 16 bits including sign, then the digitized analog value is entered in the interface starting from bit 14. The unused least significant bit positions are filled with "0".

14-bit resolution

For a resolution of 14 bits including sign and a nominal range of ±10 V, the increment is:

$20 V / 2^{14} = 20 V / 16384 \approx 1.22 mV$

Bit 0 ... 1 are always "0".

12-bit resolution

For a resolution of 12 bits including sign and a nominal range of ±10 V, the increment is:

$20 V / 2^{12} = 20 V / 4096 \approx 4.88 mV$

Bit 0 ... 3 are always "0".

Representation of the maximum value for different resolutions

Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Significance of the bits	SG	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
16-bit resolution: 32767_D = $7FFF_H$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14-bit resolution: 8191_D = $1FFF_H$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
12-bit resolution: 2047_D = $7FF_H$	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

Note

The data (resolution, nominal range) of the analog input/output module used can be taken from the documentation of the particular module.

Examples

Digital representation of analog values at a resolution of 14 bits including sign and a nominal range of ±10 V.

Example 1: Analog value = 9.5 V

Digitized analog value (decimal):	9.5 V / 20 V * 16384 = 7782
Digitized analog value 14-bit (binary):	`01 1110 0110 0110`
Digitized analog value 16-bit (binary):	`0111 1001 1001 10`00
Digitized analog value 16-bit (hex):	7998_H

Example 2: Analog value = -4.12 V

Digitized analog value (decimal):	-4.12 V / 20 V * 16384 = -3375
Digitized analog value 14-bit (binary):	`11 0010 1101 0001`
Digitized analog value 16-bit (binary):	`1100 1011 0100 01`00
Digitized analog value 16-bit (hex):	$CB44_H$

1.2.5 Comparator inputs

Function

Two internal comparator input bytes, each with eight comparator inputs, are available in addition to the digital and analog NCK inputs. The signal state of the comparator inputs is generated on the basis of a comparison between the analog values present at the fast analog inputs and the threshold values parameterized in setting data.

The $A_INCO[<n>] system variable allows the signal state (i.e. the result of the comparison) of comparator input [<n>] to be scanned directly in the part program.

Applies for index <n>:

<n> = 1 ... 8	For comparator byte 1
<n> = 9 ... 16	For comparator byte 2

Terms

The terms **comparator inputs** (with index <n>; value range from <n>: 1 ... 8 or 9 ... 16) and **comparator input bits** (with index <b>; value range from <b>: 0 ... 7) are used in this description.

They are related as follows:

For <n> = 1 ... 8:	Comparator input <n> is equivalent to comparator input bit <b> = <n> - 1.
For <n> = 9 ... 16:	Comparator input <n> is equivalent to comparator input bit <b> = <n> - 9.

Example: Comparator input 1 is equivalent to comparator input bit 0.

Assignment of the analog inputs

The following machine data is used to assign an analog input to input bit <b> of comparator byte 1:

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[<b>]

Example:

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[0] = 1

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[1] = 1

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[7] = 7

Analog input 1 acts on input bits 0 and 1 of comparator byte 1.

Analog input 7 acts on input bit 7 of comparator byte 1.

The analog inputs for comparator byte 2 are assigned with the machine data:

MD10531 $MN_COMPAR_ASSIGN_ANA_INPUT_2[<b>]

Comparator settings

The settings for the individual bits (0 to 7) of comparator byte 1 or 2 are parameterized via the machine data:

MD10540 $MN_COMPAR_TYPE_1 (parameter assignment for comparator byte 1)

or

MD10541 $MN_COMPAR_TYPE_2 (parameter assignment for comparator byte 2)

The following settings are possible:

- Comparison type mask (bits 0 … 7)

  The type of comparison condition is defined for each comparator input bit:

Bit = 1:	Associated comparator input bit is set to "1" if: Analog value ≥ threshold value
Bit = 0:	Associated comparator input bit is set to "0" if: Analog value < threshold value

- Output of the comparator input byte via digital NCK outputs (bits 16 … 23)

  The comparator bits can also be output directly via the digital NCK outputs byte-by-byte. This requires specification in this byte (bits 16 … 23) of the digital NCK output byte to be used.

Byte = 0:	No output via digital NCK outputs
Byte = 1:	Output via digital on-board-NCK outputs 9 … 16
Byte = 2:	Output via external digital NCK outputs 17 … 24
Byte = 3:	Output via external digital NCK outputs 25 … 32
Byte = 4:	Output via external digital NCK outputs 33 … 40

- Inversion mask for output of the comparator input byte (bits 24 … 31)

  For every comparator signal it is also possible to define whether the signal state to be output at the digital NCK output is to be inverted or not.

Bit = 1:	Associated comparator input bit is not inverted.
Bit = 0:	Associated comparator input bit is inverted.

Threshold values

The threshold values used for comparisons on comparator byte 1 or 2 must be stored as setting data. A separate threshold value must be entered for each comparator input bit <b> (with <b> = 0 … 7):

SD41600 $SN_COMPAR_THRESHOLD_1[<b>]

or

SD41601 $SN_COMPAR_THRESHOLD_2[<b>]

Comparator signals as digital NCK inputs

All NC functions that are processed as a function of digital NCK inputs can also be controlled by the signal states of the comparators. The byte address for comparator byte 1 (HW byte 128) or 2 (HW byte 129) must be entered in the machine data associated with the NC function ("Assignment of hardware byte used").

Example:

"Multiple feedrates in one block" NC function

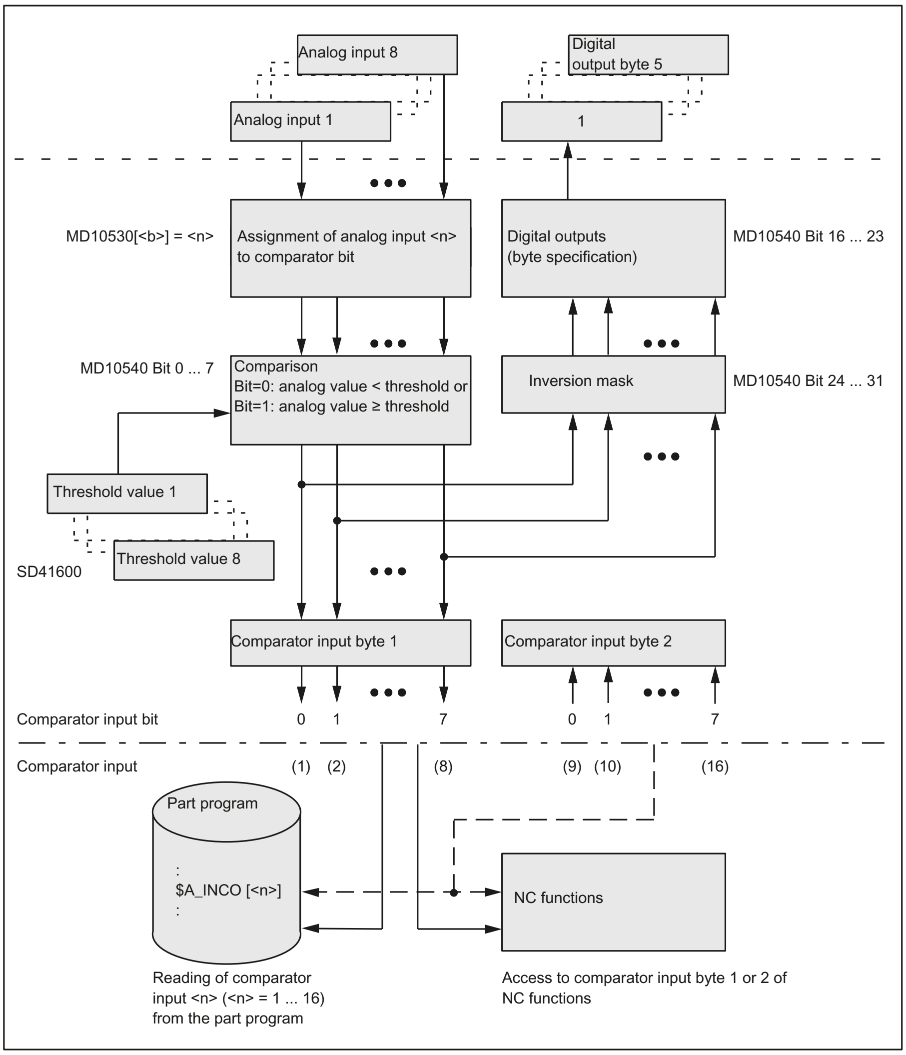
Entry in channel-specific machine data:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN = 129

This activates various feedrate values as a function of the status of comparator byte 2.

Functional sequence

The functional sequence for comparator input byte 1 is represented schematically in the following figure.



1.2.6 Direct PLC I/Os, addressable from the NC

1.2.6.1 Function

The **fast data channel** between the NCK and PLC I/Os is processed directly and therefore quickly by the PLC operating system.

There is no provision for control of the PLC basic and user programs. It is not advisable for the NCK and the PLC to attempt to access the same PLC I/Os simultaneously, as this can result in faults.

Data exchange

For access purposes, the NC uses system variables associated with part programs and synchronized actions.

Table 1- 2 For **reading** from PLC:

- $A_PBB_IN[<n>] Read byte (8-bit)
- $A_PBW_IN[<n>] Read word (16-bit)
- $A_PBD_IN[<n>] Read double word (32-bit)
- $A_PBR_IN[<n>] Read real (32-bit float)

<n> = Byte offset within the PLC input area

Reading from the part program causes a preprocessing stop.

Table 1- 3 For **writing** to PLC:

- $A_PBB_OUT[<n>] ; Write byte (8-bit)
- $A_PBW_OUT[<n>] ; Write word (16-bit)
- $A_PBD_OUT[<n>] ; Write data word (32-bit)
- $A_PBR_OUT[<n>] ; Write real (32-bit float)

<n> = Byte offset within the PLC output area

The output data can also be read from the part program and synchronized actions. Reading from the part program causes an automatic preprocessing stop (to achieve synchronization with the real-time context).

Ranges of values

Values within the following ranges can be stored in the system variables:

System variable	Value range
$A_PBB_OUT[<n>]:	(-128 ... +127) or (0 ... 255)
$A_PBW_OUT[<n>]:	(-32768 ... +32767) or (0 ... 65535)
$A_PBD_OUT[<n>]:	(-2147483648 ... +2147483647) or (0 ... 4294967295)
$A_PBR_OUT[<n>]:	(-3.402823466E+38 ... +3.402823466E+38)

Transfer times

Data is output from NCK ⇒ PLC (write) at the end of the interpolation cycle if at least one data item was written.

Data is read in by transmitting a request at the end of the interpolation cycle.

The new data is available in the subsequent interpolation cycle at the earliest.

The time period within which a request is sent to the PLC, can be specified via the machine data:

MD10398 $MN_PLCIO_IN_UPDATE_TIME

The entered time period is rounded up internally to the next highest multiple of an interpolation cycle. If the value of this machine data is set to "0", the request will continue to be sent to the PLC in every interpolation cycle.

Configuring

To activate the functionality, the following machine data (Power ON active) must be configured on the NC:

- MD10394 $MN_PLCIO_NUM_BYTES_IN

  Number of PLC I/O input bytes that are read directly by the NC
- MD10395 $MN_PLCIO_LOGIC_ADDRESS_IN

  Logical start address of the PLC input I/O, starting at which the data is read
- MD10396 $MN_PLCIO_NUM_BYTES_OUT

  Number of PLC I/O output bytes that are written directly by the NC
- MD10397 $MN_PLCIO_LOGIC_ADDRESS_OUT

  Logical start address of the PLC output I/O, starting at which the data is written
- MD10398 $MN_PLCIO_IN_UPDATE_TIME

  Time period within which the data that can be read by means of $A_PBx_IN is updated (see Section "Transfer times").
- MD10399 $MN_PLCIO_TYPE_REPRESENTATION

  Format display of system variables $A_PBx_OUT and $A_PBx_IN (see Section "Memory organization").

Note

The logical PLC I/O addresses entered in the machine data and the number of bytes to be transferred must be consistent with the PLC hardware configuration. In the configured areas, there must not be any "address gaps" in the PLC I/O expanded configuration.

Memory organization

16 bytes (over all channels) are available for data exchange from and to the PLC respectively. These areas have to be managed by the user (i.e. no overlapping of the variables, not even across channel borders).

The display of the variables within these areas depends on the setting in the machine data:

MD10399 $MN_PLCIO_TYPE_REPRESENTATION

Value	Meaning
0	(default setting) System variables are displayed in little-endian format (i.e. least significant byte at least significant address)
1	(PLC standard format, recommended) System variables are displayed in big-endian format (i.e. most significant byte at least significant address)

Since big-endian format is generally the most common display type on the PLC (i.e. it also applies to the PLC I/Os), it should normally be used.

Note

- The assignment of the input and output areas for direct PLC I/Os must satisfy the following conditions:

```
$A_PBB_IN[<j>]      ; j < ([MD10394 $MN_PLCIO_NUM_BYTES_IN])
$A_PBW_IN[<j>]      ; j < ([MD10394 $MN_PLCIO_NUM_BYTES_IN] - 1)
$A_PBD_IN[<j>]      ; j < ([MD10394 $MN_PLCIO_NUM_BYTES_IN] - 3)
$A_PBR_IN[<j>]      ; j < ([MD10394 $MN_PLCIO_NUM_BYTES_IN] - 3)

$A_PBB_OUT[<k>]     ; k < ([MD10396 $MN_PLCIO_NUM_BYTES_OUT])
$A_PBW_OUT[<k>]     ; k < ([MD10396 $MN_PLCIO_NUM_BYTES_OUT] - 1)
$A_PBD_OUT[<k>]     ; k < ([MD10396 $MN_PLCIO_NUM_BYTES_OUT] - 3)
$A_PBR_OUT[<k>]     ; k < ([MD10396 $MN_PLCIO_NUM_BYTES_OUT] - 3)
```

- The maximum number of bytes available for data exchange must not be exceeded.

1.2.6.2 Supplementary conditions

Configuration

- If the PLC I/Os are to be written/read via the fast data channel, they must always be configured as a cohesive block (i.e. no address gaps within this block).
- It must be possible for the number of bytes that have to be transferred to be mapped without gaps on the PLC I/Os.

Time response

The time when the data is read in from the PLC I/Os is not synchronized with the time when the data is made available to the part program via system variables!

Data transfer (NCK ↔ PLC)

- The data buffer is always output complete to the PLC I/Os, even if only one system variable was assigned within the data buffer.
- If values are assigned to several system variables simultaneously (e.g. for initializing the PLC I/Os), there is no guarantee that they will be transferred in the same interpolation cycle.

1.2.6.3 Examples

Writing to PLC-I/Os

The following assumptions are made in this example:

- Data is to be output directly to the following PLC I/Os:

- log. addr. 521:	8-bit digital output module
- log. addr. 522:	16-bit digital output module

- $A_PBx_OUT is used to output the data from synchronized actions.

Parameter assignment

The machine data should be set as follows:

MD10397 $MN_PLCIO_LOGIC_ADRESS_OUT= 521	; Data is output from log. addr. 521 onwards
MD10396 $MN_PLCIO_NUM_BYTES_OUT= 3	; A total of three bytes have to be output
MD10399 $MN_PLCIO_TYPE_REPRESENTATION = 1	; Data is displayed in big-endian format

Run-up of NCK and PLC

Once the NCK and PLC have run up, there is **no** cyclic data transfer to the PLC I/Os (for write access).

Programming

Loading and starting of the part program with the following content:

Program code	Comment
...	
ID=1 WHENEVER TRUE DO $A_PBB_OUT[0]=123	; Cyclic output (per interpolation cycle)
...	
ID=2 WHEN $AA_IW[x] >= 5 DO $A_PBW_OUT[1]='Habcd'	; Output of a HEX value
...	

Reading from PLC-I/Os

The following assumptions are made in this example:

- PLC I/Os:

- log. addr. 420:	16-bit analog input module
- log. addr. 422:	32-bit digital input module
- log. addr. 426:	32-bit DP slave input
- log. addr. 430:	8-bit digital input module

- $A_PBx_IN is used to read in data from a part program into R parameters.
- In order to avoid slowing down the PLC user program unnecessarily (OB1), an update time (for read access) was configured via MD10398 $MN_PLCIO_IN_UPDATE_TIME so that an update is only performed in every third interpolation cycle.

Parameter assignment

The machine data should be set as follows:

MD10395 $MN_PLCIO_LOGIC_ADRESS_IN = 420	; Data is read in from log. addr. 420 onwards
MD10394 $MN_PLCIO_NUM_BYTES_IN = 11	; A total of 11 bytes have to be read in
MD10398 $MN_PLCIO_IN_UPDATE_TIME = 0.03	; Update time period = 30 ms (interpolation cycle = 12 ms)
MD10399 $MN_PLCIO_TYPE_REPRESENTATION = 1	; Data is displayed in big-endian format

Run-up of NCK and PLC:

The update (for read access) is now performed in every third interpolation cycle after the NCK and PLC have run up.

Programming

Loading and starting of the part program with the following content:

Program code	Comment
...	
R1=$A_PBW_IN[0]	; Read in 16-bit integer
R2=$A_PBD_IN[2]	; Read in 32-bit integer
R3=$A_PBR_IN[6]	; Read in 32-bit float
R4=$A_PBB_IN[10]	; Read in 8-bit integer
...	

1.3 Access via PROFIBUS

1.3.1 Brief description

Function

The function "I/O access via PROFIBUS" implements a direct data exchange between the NCK and the PROFIBUS I/O.

Availability

The function is available for isochronous and non-isochronous configured PROFIBUS I/O.

Hardware requirements

- The required PROFIBUS I/O must be available and ready to use.
- A correct S7 HW configuration (PLC-side) must be carried out with the required PROFIBUS I/O, and loaded in the PLC.
- An I/O range must be present on the same PROFIBUS slave.

I/O range

If the individual user data slots of a PROFIBUS slave are configured in the S7 HW configuration (PLC) in such a way that they form a contiguous PROFIBUS I/O section in series, with logical start addresses in ascending order, then this section is referred to as I/O range in the following.

An I/O range is characterized by:

- A logical start address (this corresponds to the logical start address of the first user data slot of this I/O range).
- A configured length (this corresponds to the length in bytes of the connected PROFIBUS I/O to be accessed).

The logical start addresses of the I/O range must be available in the NCK, so that it can read/write the corresponding data of the PROFIBUS I/O via an NCK-internal PROFIBUS communication interface. The configured I/O range is registered via machine data (see "Configuration of the I/O ranges (Page 56)". The communication with the PROFIBUS I/O is thus possible only I/O range-oriented.

Data exchange

The data exchange with the PROFIBUS I/O is performed via an NCK-internal PROFIBUS-communication interface. The following options of data exchange with the PROFIBUS I/O are available to the NCK user:

- Reading/writing of NCK system variables ($A_DPx_IN[n,m] or $A_DPx_OUT[n,m]) via part programs / synchronized actions in the IPO cycle (data consistency). The PROFIBUS I/O data to be written is output at the PROFIBUS I/O only after the corresponding IPO cycle.

  See "Communication via part programs/synchronous actions (Page 58)".

- Read in / output of the data blocks via the compile-cycle interface (data consistency for servo cycle).

  See "Communication via compile cycles (Page 61)".

Parallel data access

A **parallel read access** through compile cycles and part programs / synchronized actions on data of the same I/O range is **possible**, as long as the corresponding I/O range has been configured for this. One must ensure that the read accesses have access to the different mappings of the PROFIBUS I/O data. The data consistency within these mappings is ensured. However, the data equality between these mappings cannot be ensured during an IPO cycle.

A **parallel write access** through compile cycles and part programs / synchronized actions on data of the same I/O range is **not possible**. While configuring the NCK it must be determined, whether a specific I/O range of the PROFIBUS I/O is allocated to the system variables or to the compile cycles.

Activation

The function is activated during the NCK run-up.

1.3.2 Configuration of the I/O ranges

The configuration of the I/O range is performed via the machine data. The parameters once set can no longer be changed during the normal operation of the NCK.

16 I/O ranges in the read direction and 16 I/O ranges in the write direction are provided. NCK restricts the maximum size of the I/O ranges to 128 bytes each.

Once an I/O range is activated, its availability is checked each time an IPO cycle starts. For this, the sign of life of a user data slot within an I/O range is evaluated. If the sign of life is not set at the start of an IPO cycle, an alarm (9050 or 9052) is issued. This alarm does not stop the processing of the part program, but instead is only displayed and remains as such till the affected I/O range gets a valid sign of life again.

Logical start address

To make specific I/O ranges available again, their logical start addresses must be conveyed to the NCK. The following machine data must be set:

MD10500 $MN_DPIO_LOGIC_ADDRESS_IN[<n>]

MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[<n>]

<n> = index for the I/O range

Length of an I/O range

In order that the NCK can check, whether an I/O range has been configured completely, the expected length (bytes) of the corresponding I/O range must be entered. The following machine data must be set:

MD10501 $MN_DPIO_RANGE_LENGTH_IN[<n>]

MD10511 $MN_DPIO_RANGE_LENGTH_OUT[<n>]

If the length "0" is entered, only the user data slot found under the corresponding logical start address is configured as I/O range. In such a case, the length of the I/O range is then compared to the length of the user data slot found.

Further attributes

Further attributes can be allocated to each I/O range with the following machine data:

MD10502 $MN_DPIO_RANGE_ATTRIBUTE_IN[<n>]

Bit	Value	Meaning
0		Format display of system variables $A_DPx_IN[<n>,<m>]
	0	Little-endian format
	1	Big-endian format
1		Reserved
2		Reading of input data
	0	Reading possible via system variables and CC-binding
	1	Reading possible only for CC-binding
3		Output of slot sign-of-life alarms
	0	Slot sign-of-life alarms are issued
	1	Slot sign-of-life alarms are suppressed

MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT[<n>]

Bit	Value	Meaning
0		Format display of system variables $A_DPx_OUT[<n>,<m>]
	0	Little-endian format
	1	Big-endian format
1		Writing of output data
	0	Writing only via system variable
	1	Writing only via CC-binding
2		Reserved
3		Output of slot sign-of-life alarms
	0	Slot sign-of-life alarms are issued
	1	Slot sign-of-life alarms are suppressed

Note

- The configuration of the I/O ranges via the corresponding machine data need not be done persistently. That is, the assignment of the I/O ranges to the corresponding machine data-indices can be selected in any way.
- If the registration of one/several I/O range(s) does not run successfully during the NCK run-up phase, the registration is aborted with an alarm (4700 or 4702).

1.3.3 Data exchange

The following requirements must be satisfied for direct data exchange with the PROFIBUS I/O via the NCK-internal PROFIBUS communication interface:

- Correct configuration of the corresponding I/O ranges.
- PLC must actually be able to provide the required I/O ranges (user data slots).
- The configured I/O ranges are released for use only when the PROFIBUS communication interface is able to do a data exchange with the corresponding PROFIBUS I/O for the first time.

1.3.3.1 Communication via part programs/synchronous actions

General

The NCK global system variables enable access to the PROFIBUS I/O, whether read or write, from the part programs / synchronized actions:

- $A_DPx_IN [<n>,<m>]
- $A_DPx_OUT [<n>,<m>]

The following must be observed:

- When reading from or writing to these variables from a part program, a preprocessing stop is triggered.
- To ensure data consistency during programming from the part program and the synchronized actions, the PROFIBUS I/O data is accessed which is kept consistent for the respective IPO cycle.
- If the same PROFIBUS I/O data is to be write-accessed several times within an IPO cycle (e.g. synchronized actions, access from different channels, etc.), then the data of the last write access is valid.
- The PROFIBUS I/O data to be written is output at the PROFIBUS I/O only after the corresponding IPO cycle.

Access I/O range data

The following system variables are available for accessing the I/O range data:

Table 1- 4 NCK → PROFIBUS I/O

System variable	Value	Meaning
$A_DPB_OUT[<n>,<m>]	8-bit unsigned	Writing a data byte (8-bit) to PROFIBUS I/O
$A_DPW_OUT[<n>,<m>]	16-bit unsigned	Writing a data word (16-bit) to PROFIBUS I/O
$A_DPSB_OUT[<n>,<m>]	8-bit signed	Writing a data byte (8-bit) to PROFIBUS I/O
$A_DPSW_OUT[<n>,<m>]	16-bit signed	Writing a data word (16-bit) to PROFIBUS I/O
$A_DPSD_OUT[<n>,<m>]	32-bit signed	Writing a data double word (32-bit) to PROFIBUS I/O
$A_DPR_OUT[<n>,<m>]	32-bit REAL	Writing output data (32-bit REAL) to PROFIBUS I/O

<n> = index for the output data range; <m> = byte index for the data

Table 1- 5 PROFIBUS I/O → NCK

System variable	Value	Meaning
$A_DPB_IN[<n>,<m>]	8-bit unsigned	Reading a data byte (8-bit) from PROFIBUS I/O
$A_DPW_IN[<n>,<m>]	16-bit unsigned	Reading a data word (16-bit) from PROFIBUS I/O
$A_DPSB_IN[<n>,<m>]	8-bit signed	Reading a data byte (8-bit) from PROFIBUS I/O
$A_DPSW_IN[<n>,<m>]	16-bit signed	Reading a data word (16-bit) from PROFIBUS I/O
$A_DPSD_IN[<n>,<m>]	32-bit signed	Reading a data double word (32-bit) from PROFIBUS I/O
$A_DPR_IN[<n>,<m>]	32-bit REAL	Reading of output data (32-bit REAL) from PROFIBUS I/O

<n> = index for the input data range; <m> = byte index for the data

Check configuration of the I/O ranges

The configuration of the I/O ranges can be checked via the following system variables. Each bit of these bit arrays corresponds to an I/O range. It is set when the I/O range is configured for access via the part programs / synchronized actions.

System variable	Value	Meaning
$A_DP_IN_CONF	32-bit bit array	Reading all configured input data ranges of the PROFIBUS I/O
$A_DP_OUT_CONF	32-bit bit array	Reading all configured output data ranges of the PROFIBUS I/O

Check availability of the I/O ranges

The availability of the I/O ranges can be checked via the following system variables. Each bit of these bit arrays corresponds to an I/O range. It is set, when the I/O-range is ready for access via the part programs/synchronous actions.

System variable	Value	Meaning
$A_DP_IN_VALID	32-bit bit array	Reading all valid input data ranges of the PROFIBUS I/O
$A_DP_OUT_VALID	32-bit bit array	Reading all valid output data ranges of the PROFIBUS I/O

Query state of an I/O range

The exact status of an I/O range can be queried with the help of the following system variables.

System variable	Value	Meaning
$A_DP_IN_STATE[<n>] <n> = index for the input data range	0: Data range was not configured 1: Data range could not be activated 2: Data range is available 3: Data range is currently not available	Reading the state of the input data range
$A_DP_OUT_STATE[<n>] <n> = index for the output data range		Reading the state of the output data range

Query length of an I/O range

The configured length of an I/O range can be queried using the following system variables.

System variable	Meaning
$A_DP_IN_LENGTH[<n>] <n> = index for the input data range	Reading the length of the input data range
$A_DP_OUT_LENGTH[<n>] <n> = index for the output data range	Reading the length of the output data range

Note

- Via <n> (RangeIndex) the corresponding NCK-configured I/O range is selected. If the required I/O range is not configured, it is indicated by issuing an alarm (17020).
- The <m> (RangeOffset) points to the place (byte-offset) within the I/O range, from which the data access is to be started. Data types can be read/written at any byte offset within the I/O range. Read/write accesses which exceed the configured limits of the respective I/O range are rejected with the generation of an alarm (17030).
- Via the machine data MD10502 $MN_DPIO_RANGE_ATTRIBUTE_IN or MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT (see "Configuration of the I/O ranges (Page 56)"), the display format (little-/big-endian) for $A_DPx_IN[n,m] or $A_DPx_OUT[n,m] system variables can be defined for the read/write direction as well as for each individual I/O range.

1.3.3.2 Communication via compile cycles

General

CC-bindings are available for importing/exporting data blocks via the compile cycle interfaces. The access to the data of the I/O range takes place at the servo task level. The data is updated in each servo cycle. Data consistency is thus given for each respective servo cycle.

To have write access to the data of the I/O range via the CC-bindings, the relevant I/O ranges must be activated during the NCK configuration for the programming via compile cycles:

MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT[<n>], bit 1 = 1

A simultaneous programming of these I/O ranges via part programs / synchronized actions is prevented by issuing an alarm (17020).

It must be noted that the data is displayed in general in the PLC in the big-endian format. Naturally, this also applies to the PROFIBUS I/O. Since the bindings support only byte-oriented access to data ranges (byte offset, number of bytes to be transferred) within an I/O range, you must pay attention to the correct display of the data types (16-bit, 32-bit, etc.).

CC-Bindings

The following CC-bindings are available:

CCDataOpi: getDpIoRangeConfiguration()

CCDataOpi: getDpIoRangeValid()

CCDataOpi: getDpIoRangeInInformation()

CCDataOpi: getDpIoRangeOutInformation()

CCDataOpi: getDpIoRangeInState()

CCDataOpi: getDpIoRangeOutState()

CCDataOpi: getDataFromDpIoRangeIn()

CCDataOpi: putDataToDpIoRangeOut()

Note

- The bindings
  CCDataOpi: getDataFromDpIoRangeIn() or
  CCDataOpi: putDataToDpIoRangeOut()
  monitor during the read/write accesses the adherence to the limits of the respective I/O range configured at the NCK and PLC side. Access to data / data ranges which do not lie completely within the configured I/O range limits is rejected by returning the enumerator CCDATASTATUS_RANGE_LENGTH_LIMIT.
- If an attempt is made to access an I/O range which is not configured (or not configured for the compile cycle), it is notified by means of the return enumerator CCDATASTATUS_RANGE_NOT_AVAILABLE.

> NOTICE
>
> **Possible system slowdown**
>
> The compile cycle programmer is responsible for the correct use of the CC-bindings! It must be noted that the additional performance requirement needed for providing the data of the configured I/O ranges at the servo task level, does not lead to a servo level computing time overflow.
>
> See the OEM documents for more information about the use of these bindings.

1.3.4 Supplementary conditions

Hardware

- Only the PROFIBUS-slaves at the first real PROFIBUS line of the PLC (plug with the identification DP1) are supported.

Software

- A simultaneous writing of the PROFIBUS I/O from a PLC-application program is not permitted and cannot be prevented technically.
- I/O ranges for the write access (MD10510 $MN_DPIO_LOCIG_ADDRESS_OUT[]) to the PROFIBUS I/O may not lie in the I/O mapping range of the PLC (e.g. PLC 317, addresses from 0 - 255), since this range is used by the PLC operating system.

1.3.5 Examples

1.3.5.1 PROFIBUS-I/O in write direction

Requirement

The S7-HW-configuration is already done.

Configuration for programming via part program/synchronous actions

- RangeIndex = 5 (NCK-internal configuration)
- as per S7-HW-configuration:
  - log. start address = 334
  - Slot length = 8 byte
  - To be displayed in Little-Endian format

This results in the following configuration of the machine data:

MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[5] = 334 (log. start address I/O-range)

MD10511 $MN_DPIO_RANGE_LENGTH_OUT[5] = 8 (length of the I/O-range)

MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT[5]
Bit0 = 0 (Little-Endian-Format)
Bit1 = 0 (writing only via system variable)
Bit3 = 0 (Slot-lifespan-alarms issued)

Configuration for programming via CompileCycles

- RangeIndex = 6 (NCK-internal configuration)
- as per S7-HW-configuration:
  - log. start address = 444
  - Slot length = 10 byte
  - To be displayed in Little-Endian format

This results in the following configuration of the machine data:

MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[6] = 444 (log. start address I/O-range)

MD10511 $MN_DPIO_RANGE_LENGTH_OUT[6] = 0
(a single useful-data slot is to be used)

MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT[6]
Bit0 = 0 (Little-Endian-Format)
Bit1 = 1 (writing only via CC-binding)
Bit3 = 1 (Slot-lifespan-alarms suppressed)

Programming

Program code	Comment
$A_DPB_OUT[5,6]=128	; Write (8 bit) to RangeIndex=5, RangeOffset=6
$A_DPW_OUT[5,5]='B0110'	; Write (16 bit) to RangeIndex=5, RangeOffset=5
	; Little Endian format
	; Notice: RangeData of byte 6 are overwritten
$A_DPSD_OUT[5,3]='8FHex'	; Write (32 bit) to RangeIndex=5, RangeOffset=3
	; Little Endian format
	; Notice: RangeData of byte 5.6 are overwritten
$AC_MARKER[0]=5	
$AC_MARKER[1]=3	
$A_DPSD_OUT[$AC_MARKER[0],$AC_MARKER[1]]='8FHex'	
	; Write (32 bit) to RangeIndex=5, RangeOffset=3
	; Little Endian format
	; indirect programming
R1=$A_DPB_OUT[5,6]	; Read (8 bit) to RangeIndex=5, RangeOffset=6
	; Little Endian format
	; Result: 0xFF
ID=1 WHENEVER TRUE DO $A_DPB_OUT[5,0]=123	
	; Cyclic output (per IPO cycle)
$A_DPB_OUT[5.255]=128	; Alarm 17030 because the RangeOffset is invalid.
$A_DPB_OUT[6.10]=128	; Alarm 17020 because this range of the part program
	; cannot be written.
$A_DPB_OUT[7.10]=128	; Alarm 17020 because this range is not defined.
$A_DPB_OUT[16.6]=128	; Alarm 17020 because RangeIndex >= max. available
	; number of ranges.

1.3.5.2 PROFIBUS-I/O in read direction

Requirement

The S7-HW-configuration is already done.

Configuration for programming via part program/synchronous actions

- RangeIndex = 0 (NCK-internal configuration)
- as per S7-HW-configuration:
  - log. start address = 456
  - Slot length = 32 byte
  - To be displayed in Big-Endian format

This results in the following configuration of the machine data:

MD10500 $MN_DPIO_LOGIC_ADDRESS_IN[0] = 456 (log. start address I/O-range)

MD10501 $MN_DPIO_RANGE_LENGTH_IN[0] = 32 (length of the I/O-range)

MD10502 $MN_DPIO_RANGE_ATTRIBUTE_IN[0]
Bit0 = 1 (Big-Endian-Format)
Bit2 = 0 (read possible via system variable and CC-binding)
Bit3 = 0 (Slot-lifespan-alarms issued)

Configuration for programming via CompileCycles

- RangeIndex = 1 (NCK-internal configuration)
- as per S7-HW-configuration:
  - log. start address = 312
  - Slot length = 32 byte
  - To be displayed in Little-Endian format

This results in the following configuration of the machine data:

MD10500 $MN_DPIO_LOGIC_ADDRESS_IN[1] = 312 (log. start address I/O-range)

MD10501 $MN_DPIO_RANGE_LENGTH_IN[1] = 32 (length of the I/O-range)

MD10502 $MN_DPIO_RANGE_ATTRIBUTE_IN[1]
Bit0 = 1 (Big-Endian-Format)
Bit2 = 1 (read possible only via CC-binding)
Bit3 = 1 (Slot-lifespan-alarms suppressed)

Programming

Program code	Comment
$AC_MARKER[0]=$A_DPW_IN[0,0]	; Read (16 bit) to RangeIndex=0, RangeOffset=0
	; Big Endian format
$AC_MARKER[1]=$A_DPSD_IN[0,1]	; Read (32 bit) to RangeIndex=0, RangeOffset=1
	; Big Endian format
$AC_MARKER[1]=$A_DPSD_IN[0.8]	; Read (32 bit) to RangeIndex=0, RangeOffset=8
	; Big Endian format
$AC_MARKER[2]=0	
$AC_MARKER[3]=8	
$AC_MARKER[1]=$A_DPSD_IN[$AC_MARKER[2],$AC_MARKER[3]]	
	; Read (32 bit) to RangeIndex=0, RangeOffset=8
	; Big Endian format
	; indirect programming
ID=2 WHEN $A_DPB_IN[0,11]>=5 DO $AC_MARKER[2]='ABCDHex'	
	; Cyclic fetching (per IPO cycle)
R1=$A_DPB_IN[0,255]	; Alarm 17030 because the RangeOffset is invalid.
R1=$A_DPB_IN[2.6]	; Alarm 17020 because this range is not defined.
R1=$A_DPB_IN[1.10]	; Alarm 17020 because this range of the part program cannot be written.
R1=$A_DPB_IN[16.6]	; Alarm 17020 because RangeIndex >= max. available
	; number of ranges.

1.3.5.3 Query of the RangeIndex in case of "PROFIBUS-I/O in write direction"

Requirement

The S7-HW-configuration is already done.

Configuration for programming via part program/synchronous actions

- RangeIndex = 5 (NCK-internal configuration)
- as per S7-HW-configuration:
  - log. start address = 1200
  - Slot length = 32 byte
  - To be displayed in Big-Endian format

This results in the following configuration of the machine data:

MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[5] = 1200 (log. start address I/O-range)

MD10511 $MN_DPIO_RANGE_LENGTH_OUT[5] = 0
(a single useful-data slot is to be used)

MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT[5]
Bit0 = 1 (Big-Endian-Format)
Bit1 = 0 (writing only via system variable)
Bit3 = 0 (Slot-lifespan-alarms issued)

Programming

before an access query the status of RangeIndex = 5

```
N3     check:                                  ; Jump marker
N5     IF $A_DP_OUT_STATE[5]==2 GOTOF write    ; if data range valid
                                               ; => jump to N15
N10    GOTOB check                             ; jump back to check
N15    write:                                  ; Jump marker
N20    $A_DPB_OUT[5,6]=128                     ; Writing the data byte
```

Query, whether all configured ranges/slots are valid

```
N3     check:                                  ; Jump marker
N5     IF $A_DP_OUT_CONF==$A_DP_OUT_VALID GOTOF ; if data range valid
       write                                   ; => jump to N15
N10    SETAL(61000)                            ;Set alarm no. 61000
N15    write:                                  ; Jump marker
N20    $A_DPB_OUT[5,6]=128                     ; Writing the data byte
```

Query, whether the configured RangeIndex = 5 is valid

```
N3     check:                                  ; Jump marker
N5     IF $A_DP_OUT_VALID B_AND 'B100000' GOTOF ; if data range valid
       write                                   ; => jump to N15
N10    SETAL(61000)                            ;Set alarm no. 61000
N15    write:                                  ; Jump marker
N20    $A_DPB_OUT[5,6]=128                     ; Writing the data byte
```

Querying the length of the configured, valid I/O-range with RangeIndex = 5

```
N100   R1=$A_DP_OUT_LENGTH[5]      ; Length of the I/O-range (slot) in bytes
                                   ;Result: R1 = 32
```

1.4 Data lists

1.4.1 Machine data

1.4.1.1 General machine data

Number	Identifier: $MN_	Description
10300	FASTIO_ANA_NUM_INPUTS	Number of active analog NCK inputs
10310	FASTIO_ANA_NUM_OUTPUTS	Number of active analog NCK outputs
10320	FASTIO_ANA_INPUT_WEIGHT	Weighting factor for analog NCK inputs
10330	FASTIO_ANA_OUTPUT_WEIGHT	Weighting factor for analog NCK outputs
10350	FASTIO_DIG_NUM_INPUTS	Number of active digital NCK input bytes
10360	FASTIO_DIG_NUM_OUTPUTS	Number of active digital NCK output bytes
10362	HW_ASSIGN_ANA_FASTIN	Hardware assignment of external analog NCK inputs
10364	HW_ASSIGN_ANA_FASTOUT	Hardware assignment of external analog NCK outputs
10366	HW_ASSIGN_DIG_FASTIN	Hardware assignment of external digital NCK inputs
10368	HW_ASSIGN_DIG_FASTOUT	Hardware assignment of external digital NCK outputs
10394	PLCIO_NUM_BYTES_IN	Number of directly readable input bytes of the PLC I/Os
10395	PLCIO_LOGIC_ADDRESS_IN	Start address of the directly readable input bytes of the PLC I/Os
10396	PLCIO_NUM_BYTES_OUT	Number of directly writeable output bytes of the PLC I/Os
10397	PLCIO_LOGIC_ADDRESS_OUT	Start address of the directly writeable output bytes of the PLC I/Os
10398	PLCIO_IN_UPDATE_TIME	Update time for PLC I/O input cycle
10399	PLCIO_TYPE_REPRESENTATION	Little-/big-endian representation for PLC I/O
10500	DPIO_LOGIC_ADDRESS_IN	Logical slot address of the PROFIBUS I/O
10501	DPIO_RANGE_LENGTH_IN	Length of the PROFIBUS I/O range
10502	DPIO_RANGE_ATTRIBUTE_IN	Attributes of the PROFIBUS I/O
10510	DPIO_LOGIC_ADDRESS_OUT	Logical slot address of the PROFIBUS I/O
10511	DPIO_RANGE_LENGTH_OUT	Length of the PROFIBUS I/O range
10512	DPIO_RANGE_ATTRIBUTE_OUT	Attributes of the PROFIBUS I/O
10530	COMPAR_ASSIGN_ANA_INPUT_1	Hardware assignment of NCK analog inputs for comparator byte 1
10531	COMPAR_ASSIGN_ANA_INPUT_2	Hardware assignment of NCK analog inputs for comparator byte 2
10540	COMPAR_TYPE_1	Parameterization for comparator byte 1
10541	COMPAR_TYPE_2	Parameterization for comparator byte 2

1.4.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
21220	MULTFEED_ASSIGN_FASTIN	Assignment of input bytes of NCK I/Os for "multiple feedrates in one block"

1.4.2 Setting data

1.4.2.1 General setting data

Number	Identifier: $SN_	Description
41600	COMPAR_THRESHOLD_1	Threshold values for comparator byte 1
41601	COMPAR_THRESHOLD_2	Threshold values for comparator byte 2

1.4.3 Signals

1.4.3.1 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Disable digital NCK inputs	DB10.DBB0/122/124/126/128	DB2800.DBB0/1000
Setting on PLC of digital NCK inputs	DB10.DBB1/123/125/127/129	DB2800.DBB1/1001
Disable digital NCK outputs	DB10.DBB4/130/134/138/142	DB2800.DBB4/1008
Overwrite mask for digital NCK outputs	DB10.DBB5/131/135/139/143	DB2800.DBB5/1009
Setting value from PLC for the digital NCK outputs	DB10.DBB6/132/136/140/144	DB2800.DBB6/1010
Setting mask for digital NCK outputs	DB10.DBB7/133/137/141/145	DB2800.DBB7/1011
Disable analog NCK inputs	DB10.DBB146	-
Setting mask for analog NCK inputs	DB10.DBB147	-
Setting value from PLC for the analog NCK inputs	DB10.DBB148-163	-
Overwrite mask for analog NCK outputs	DB10.DBB166	-
Setting mask for analog NCK outputs	DB10.DBB167	-
Disable analog NCK outputs	DB10.DBB168	-
Setting value from PLC for the analog NCK outputs	DB10.DBB170-185	-

1.4.3.2 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Actual value for digital NCK inputs	DB10.DBB60/186-189	DB2900.DBB0/1000
Setpoint for digital NCK outputs	DB10.DBB64/190-193	DB2900.DBB4/1004
Actual value for analog NCK inputs	DB10.DBB194-209	-
Setpoint for analog NCK outputs	DB10.DBB210-225	-

B3: Distributed systems - 840D sl only

2

2.1 Brief description

2.1.1 Several operator panels on several NCUs (T:M:N)

Under certain circumstances, a single operator control and monitoring station may not be sufficient for complex plants and machines. Therefore, several operator control and monitoring stations in a SINUMERIK system network (Ethernet) can be connected to several numerical controls (NCU) via a PCU in such a way that they enable flexible and distributed operation and monitoring of the entire system.

The following figure provides an overview of the components that currently can be connected in a system network to form an operator control and monitoring system T:1:N. Meanings:

- T: Thin Client Unit (TCU) or HT8 handheld unit (connected to the PCU)
- 1 (M): Man Machine Control (MMC), PCU 50.x with SINUMERIK Operate
- N: Numeric Control Unit (NCU), NCU 7x0.3 PN

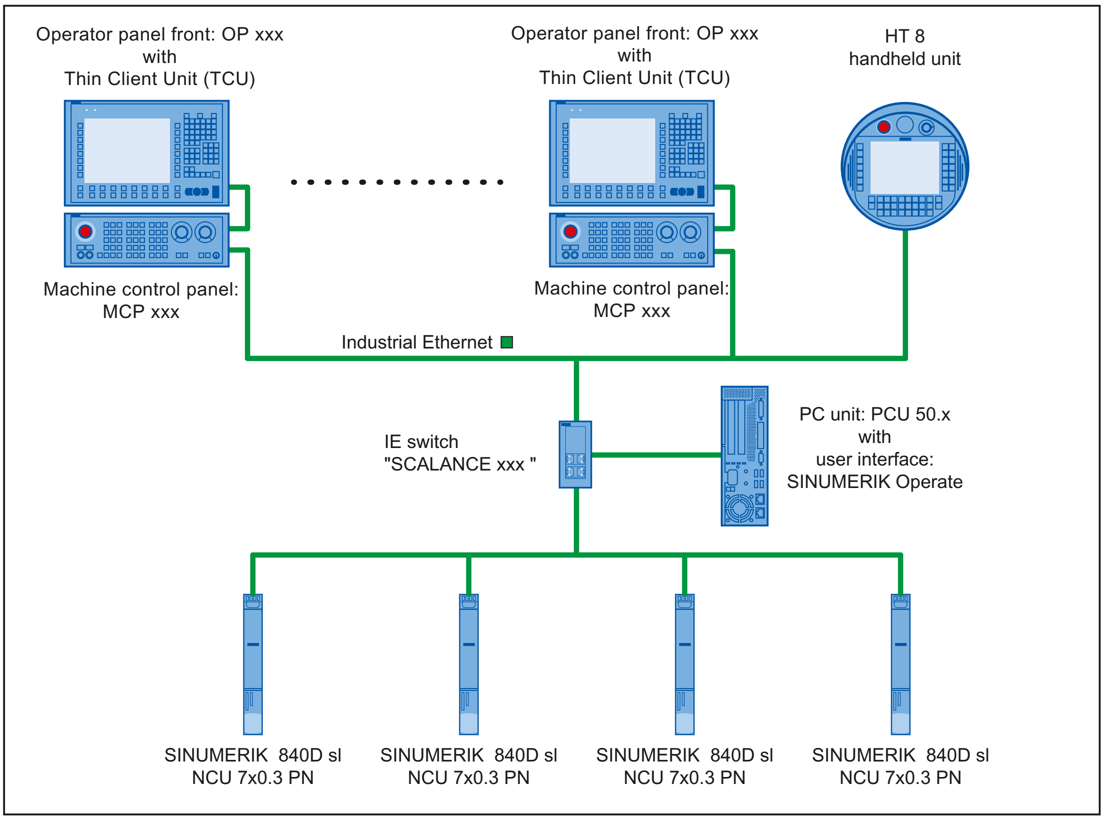


Figure 2-1 Example of a T:1:N network

Quantity structure

The following quantity structure must be observed for an operator control and monitoring system T:1:N:

T: Thin Client Unit (TCU) or HT8 handheld unit

The graphic information of the PCU 50.x user interface is transferred via the TCU to the operator panel front (OP) and displayed there. TCU and operator panel front together form an operator station.

- Maximum number

  In principle, any number of operator stations, only limited by the amount of available address space, can be operated in the system network.
- Online

  A maximum of four operator stations per PCU can be online simultaneously.
- Operating focus

  At any one time, only one of the operator stations on the PCU that are online, can have the operating focus. Entries with regard to the user interface can only be made via this operator station. The user interfaces on the other online operator stations are darkened to indicate that they do not have operating focus. Only a message is displayed on operator stations that are not online.

  The operating focus can be switched between the online operator stations.
- HT8 handheld unit

  A HT8 handheld unit combines a TCU, an operator panel front and a machine control panel in a single device.

M: PCU 50.x with SINUMERIK Operate

Several operator stations and several NCU 7x0.3 PN can be connected to a PCU 50.x with SINUMERIK Operate user interface. The data of the channels of different NCUs can be displayed on the operator stations through suitable configuring of the channel menu.

Note

Currently, only one PCU per T:1:N network can be configured ⇒ **M = 1**

N: SINUMERIK 840D sl, NCU 7x0.3 PN

- Internal SINUMERIK Operate user interface

  Because of the existing external SINUMERIK Operate user interface on PCU 50.x in the system network, the NCU-integrated internal SINUMERIK Operate user interface must be switched off. The internal SINUMERIK Operate user interface must not be switched off only when it is used with a limited range of functions, exclusively within the scope of tool management. Detailed information on this can be found in the Commissioning Manual "Operator Components and Networking (IM5)", Section "Network configuration" > "Application example"
- Machine control panel (MCP)

  For a flexible switchover of the machine control panels, only one MCP can be connected to one NCU. The switchover of the MCPs is user-specifically via the PLC user program (function block FB 9).

Installation and connection

References

- TCU, MCP, PCU:
  SINUMERIK 840D sl Operator Components and Networking Manual
- NCU:
  SINUMERIK 840D sl NCU 7x0.3 PN Manual
- Machine control panel (MCP)
  FB 9: MtoN Control Unit Switchover
- Switch SCALANCE:
  Communication with SIMATIC System Manual

Configuration, commissioning and parameter assignment

References

- Structure of the system network and commissioning of a TCU:
  Operator Components and Networking Commissioning Manual (IM5)
- Configuration of the channel menu
  SINUMERIK Operate Commissioning Manual (IM9), Section "Channel menu"

Programming

References

- MCP:
  Basic Functions Function Manual, Section "P3 Basic PLC program for SINUMERIK 840D sl" > "Block descriptions" > "FB 9: MtoN control unit switchover"

2.1.2 NCU link

2.1.2.1 Link communication

The NCU-link communication cycle allows an interpolation-synchronous cross-NCU data exchange for the following applications:

- Cross-NCU data exchange via link variable $A_DLx
- Cross-NCU traversing of axes and spindles using link axes and container-link axes
- Cross-NCU master value coupling of axes and spindles using lead-link axes

Safety Integrated

If axial Safety Integrated functions are active for a link axis, data that is not safety-related is exchanged between the local safety axis (link axis on NCUx) and the channel that is traversing it (NCUy) via the NCU link. This also enables Safety Integrated-supporting functions for link axes, such as velocity/speed monitoring-dependent setpoint limiting and higher-order stop functions (STOP D/E).

Note

The Safety Integrated functions "Safe software limit switch" (SE) and "Safe software cam, safe cam track" (SN) are not (yet) supported.

Link module

The link communication requires optional link modules. A link module is an optional PROFINET module for isochronous real-time communication (IRT) via Ethernet. The link module occupies the option slot of the NCU.

Quantity structure

As standard, a maximum of three NCUs can be interconnected to form a link group.

Note

For a specific project, on request to your local regional Siemens representative, further NCUs can be integrated to form a link group.

2.1.2.2 Link variables

Link variables are global system user variables that for configured link communication can be used by all NCUs of the network. Link variables can read and written in part programs, cycles and synchronized actions.

The reading and writing of link variables is performed in the main run. A channel synchronization is therefore performed implicitly with regard to the link variables so that all channels read the same value in the same interpolator cycle. By comparison, global user data (GUD) can also be used cross-channel like the link variables. However, as the GUD is processed in the preprocessing, no implicit channel synchronization is performed here. If necessary, GUD must therefore be synchronized by the user through specific programming.

Note

On systems without an NCU link, link variables can also be used as additional global user data alongside standard global user data (GUD).

2.1.2.3 Link axes

Link axes enable the creation of distributed control systems, in which the setpoints of the machine axes are not created by channels of the NCU to which the machine axes are physically connected, but by channels of an arbitrary NCU of the link group.

Fixed assignment for link axes

The assignment, on which NCU the setpoints for which machine axis are created, is fixed for link axes through the machine data configuration.

Variable assignment for container link axes

The assignment is variable for container link axes. A container link axis can therefore be traversed alternatively by each NCU of the link group. See Section "Axis container (Page 100)".

Irrespective of the assignment described above, the logical machine axis can be made known to the link axis of several channels on the NCU generating the setpoint and then functionally handled as a local logical machine axis, e.g. master value for position/feed couplings, NCU-local axis interchange.

2.1.2.4 Lead link axes

If in conjunction with the coupling functions of the generic coupling, all interpolators, i.e. the setpoint-creating/processing channels, the leading and following axes/spindles, are on the same NCU, the use of a lead link axis is not required. The machine axes of the leading and/or following axes/spindles can also be connected link axes to different NCUs.

Application

A lead link axis is always required when not all interpolators of a coupling group are on the same NCU. The lead link axis provides an image of the leading axis/spindle or the master value for the local interpolators, on another NCU. Starting from this master value, the setpoints of the following axes/spindles are then generated by the interpolators on this NCU.

Further application

A lead link axis is also used if more machine axes are involved in a coupling than are available absolutely or relatively on an NCU for the coupling.

References

Special Functions Function Manual, Section "Axis couplings (M3)" > "Generic coupling"

2.1.2.5 Dependencies

In order that several axes can be traversed in an interpolatary relationship, it is essential that the setpoints generated by the interpolator of the traversing channel be transferred to the position control at the same time.

During the transfer of the setpoints via the NCU link, for example, for a link axis there is a delay of one interpolator cycle in relation to the setpoint of a local axis. The delay must be compensated by a suitable increase of the buffer between the interpolator and the position controller :

MD18720 $MN_MM_SERVO_FIFO_SIZE = <value> = <default value + compensation value>

Function	Compensation value
Link axis Container-link axis	One interpolator cycle must be compensated on all NCUs with axes that interpolate with a link or container link axis. Compensation value = 1
Lead-link axis	Two interpolator cycles must be compensated on the NCU of the leading axis that generates the setpoints for the lead link axis. Compensation value = 2

Note

In contrast to lead link axes, link and container link axes require another compensation value (MD18720 $MN_MM_SERVO_FIFO_SIZE). With simultaneous use of link or container link axes and lead link axes within a link group, the delays cannot be compensated.

2.1.2.6 Application example: Rotary indexing machine

On the basis of rotary cycle machine with two NCUs, the application of the "NCU Link" function is shown as example. The principal units of the rotary indexing machine are:

- MTR rotary axis for the rotary table
- MS1 - MS4 spindles
- Machining stations 1 and 2, each with the X1 and Z1 linear axes
- Loading and unloading station

All of the machine axes remain permanently assigned to their particular NCU. The same axes (X and Z) and spindles (S1) are always addressed in the machining program of the relevant NCU.

The workpiece status is represented schematically after each machining step.

For each machining step, the rotary table (MTR) advances one position counter-clockwise. This so assigns the spindles to another station for each machining step. The changing relationships of the spindles defined in the channels to the machine axes are represented as an **axis container** in the NCU.

If the machine axis of a spindle is not contained on its own NCU, the spindle setpoints are transferred by **link communication** to the associated NCU and output there at the machine axis.

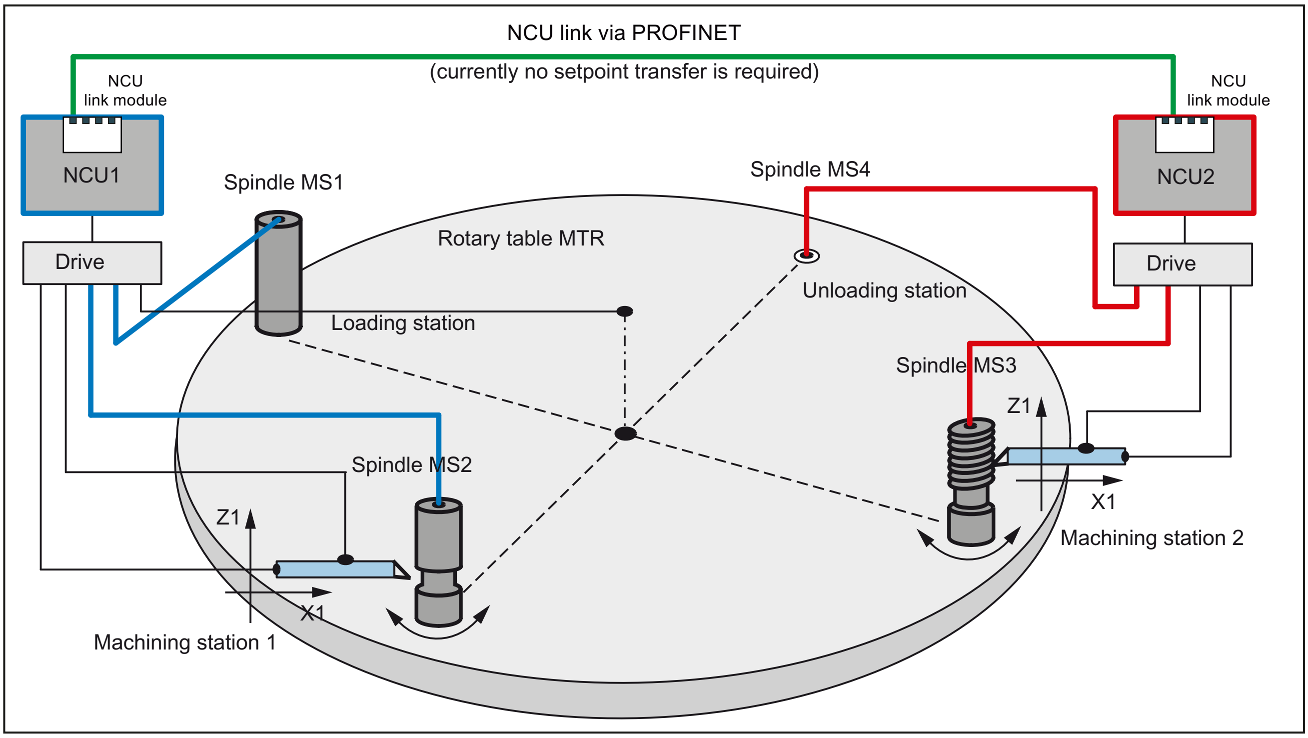


Figure 2-2 Fig. 1: Initial situation

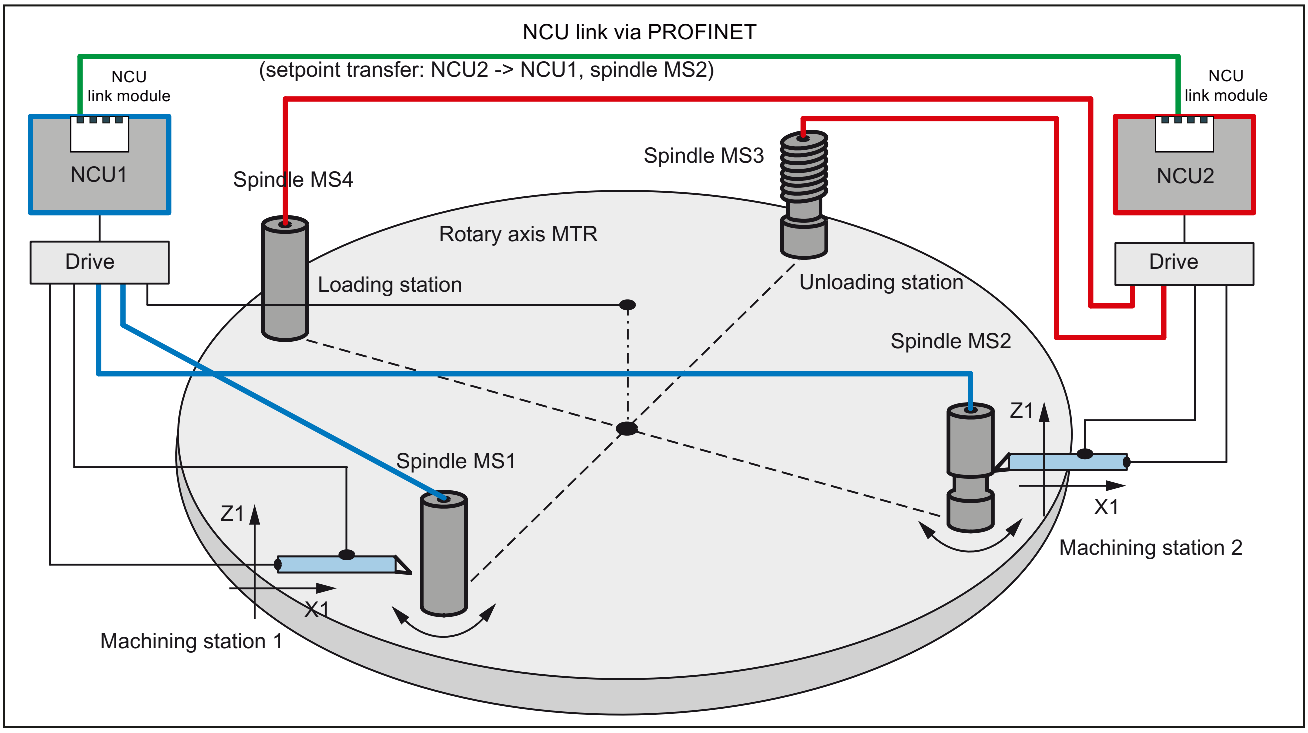


Figure 2-3 Fig. 2: Location after rotation by one position

Parameterization (schematic)

General		
Programmed channel axes in the part programs of both NCUs: X, Z, S1		
NCU1	Machine axes defined in the NCU:	
	Local:	X1, Z1
	Axis container:	MS1, MS2
NCU2	Machine axes defined in the NCU:	
	Local:	X1, Z1
	Axis container:	MS3, MS4

	Initial setting (Fig. 1)	Rotation of the MTR (rotary table) rotary axis by one position (Fig. 2)
NCU1	Machining station 1: X1, Z1, MS2	
	Diagram showing the channel axes programmed in the part program:	
	X → X1 and Z → Z1	X → X1 and Z → Z1
	S1 → MS2	S1 → MS1
NCU2	Machining station 2: X1, Z1, MS2	
	Diagram showing the channel axes programmed in the part program:	
	X → X1 and Z → Z1	X → X1 and Z → Z1
	S1 → MS3	S1 → MS2 (link axis)

2.2 NCU link

2.2.1 Link communication

2.2.1.1 General information

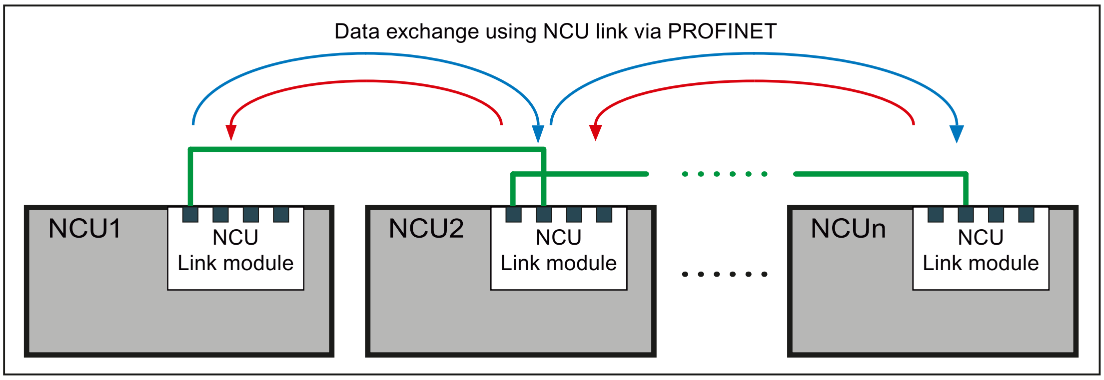


Figure 2-4 Link communication (principle)

The NCU-link communication cycle allows an interpolation-synchronous cross-NCU data exchange for the following applications:

- Cross-NCU link variable $A_DLx

  All the NCUs involved with the NCU-link communication have a common view of the link variables because they are exchanged via the NCU link interpolator cycle clock-synchronous between the NCU of the link group.

- Cross-NCU traversing of axes and spindles using link axes and container-link axes

  If single or interpolation axis traversal is made from a channel of an NCU, NCU-link setpoints can also be transferred to axes physically connected to another NCU of the link group. These axes are designated as link axes.

- Cross-NCU master value coupling of axes and spindles using lead-link axes

  NCU1 traverses axis X1 (leading axis), the setpoints are transferred to a link axis of NCU2 (lead-link axis) via the NCU link. Coupling of axis X2 is realized in NCU2 on this lead-link axis. This means axis X2 is indirectly the following axis of X1.

Data transfer

Depending on the active functions, the following cyclic and non-cyclical data transmissions occur between the NCUs involved with the link communication:

- High-priority, cyclic data transfer:
  - Setpoints, actual values and status signals of the axes
  - NCU status signals
- High-priority, non-cyclic, non-displaceable data transfer:
  - Non-safety-related data as part of the Safety Integrated function
  - States of the container axes during an axis container rotation
- Low-priority, non-cyclic, displaceable data transfer (listed in order of decreasing priority):
  - Link variables
  - Warm restart requirements
  - Activation of axis container rotations
  - Changes in NCU-global machine and setting data
  - Activation of the axial machine data for link axes
  - Alarms

Displacement

If for low-priority, non-cyclic, displaceable data transmission not all requirements can be sent in a single interpolator cycle, the request with higher priority displaces that with a lower priority. This will then be sent later.

Quantity structure: NCU 7x0.3 PN

As standard, a maximum of three NCUs can be interconnected to form a link group.

Note

For a specific project, on request to your local regional Siemens representative an NCU-link group with more than three NCUs is possible. Without project-specific supplements, more than three NCUs are rejected with Alarm 380020.

NCU link and Safety Integrated

The following figure shows a constellation with two NCUs and two machine axes, of which the MA2 machine axis of the NCU2 is traversed as link axis for NCU1. The Safety Integrated function monitors the safety-related aspects of both axes.

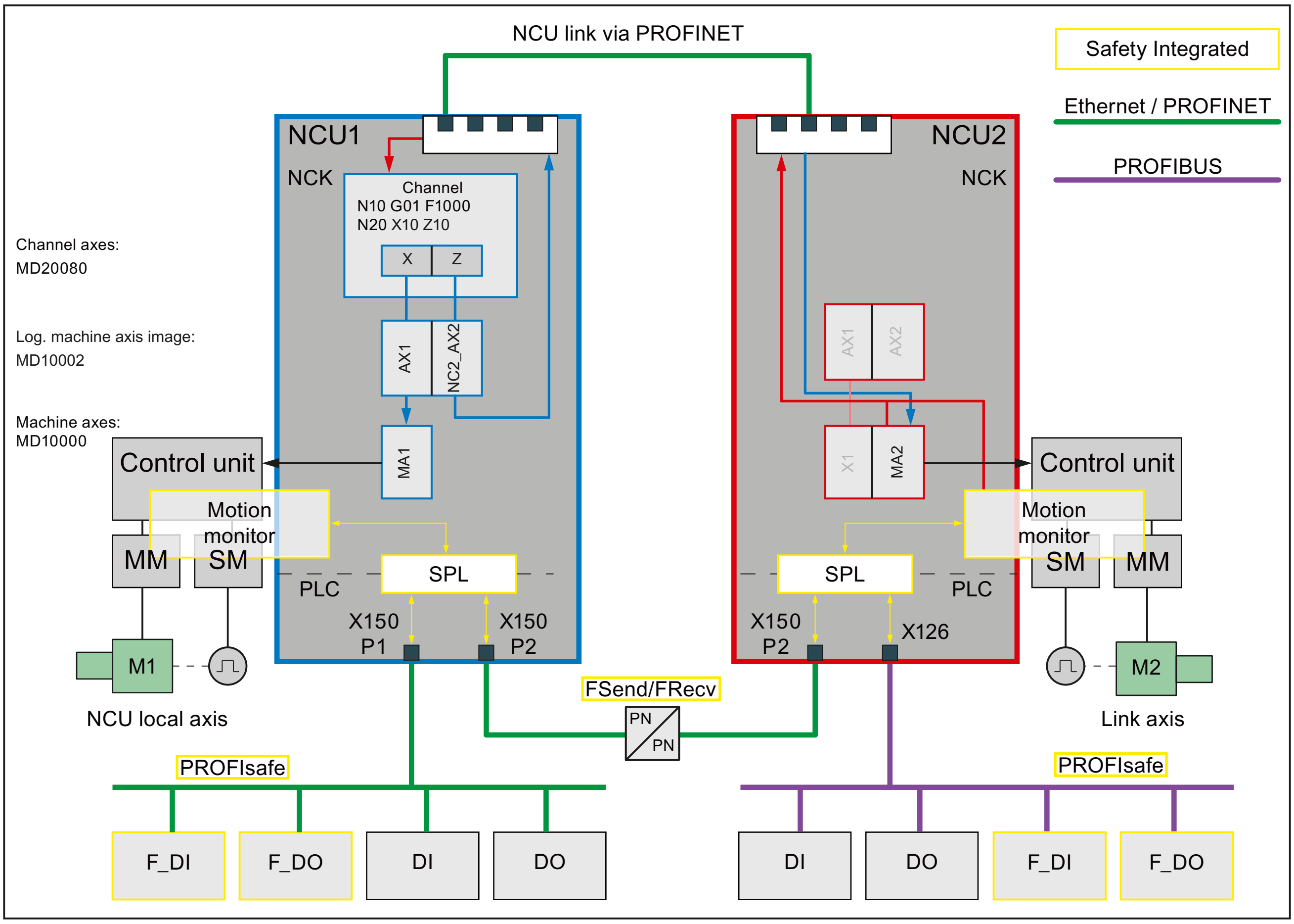


Figure 2-5 NCU link and Safety Integrated

The safety-related input signals (F_DI) of both NCUs can be acquired, linked via the safe programmable logic (SPL) and exchanged using the safety-related CPU-CPU communication (FSend/FRecv).

As part of the Safety Integrated function, the NCU-link communication permits the single-channel **non-safety-related** transfer of link axis data for Safety Integrated-supporting functions.

Examples:

- Interpolatory braking for an implicit Stop D of all traversing axes, including the link axes, in the channel.
- Reduce the speed of all axes traversing in the channel, including link axes, for a change of the safe speed.

Safety Integrated acceptance test and NCU link

The Acceptance Test wizard (ATW) is generally used to perform the acceptance test separately for each NCU. Only alarms on the home NCU of the axis are displayed for link axes. To make the ATW also check alarms for link axes, the ATW must be told the safety-relevant NCU connected via the NCU link. For this purpose, these must be entered in the NETNAMES.INI file on the PG/PCU on which the ATW runs.

Example: Entries in the NETNAMES.INI file for NCU link with two safety-relevant NCUs

```
Adapt file: NETNAMES.INI (excerpt)
[own]
owner = HMI_1
[conn HMI_1]
conn_1= NCU_1
conn_2= NCU_2
[param HMI_1]
mmc_address = 1
[param network]
[param NCU_1]
nck_address= 192.168.214.2,LINE=10,NAME=/NC,SAP=030d,PROFILE=CLT1__CP_L4_INT
plc_address= 192.168.214.2,LINE=10,NAME=/PLC,SAP=0201,PROFILE=CLT1__CP_L4_INT
ADDRESS2=    192.168.214.2,LINE=10,NAME=/CP,SAP=0501,PROFILE=CLT2__CP_L4_INT
ADDRESS10=   192.168.214.2,LINE=10,NAME=/DRIVE_00_000,SAP=0201,SUBNET=0000-
00000000:000,PROFILE=CLT2__CP_L4_INT
ADDRESS11=   192.168.214.2,LINE=10,NAME=/DRIVE_03_003,SAP=0900,PROFILE=CLT2__CP_L4_INT
ADDRESS12=   192.168.214.2,LINE=10,NAME=/DRIVE_03_011,SAP=0B00,PROFILE=CLT2__CP_L4_INT
ADDRESS13=   192.168.214.2,LINE=10,NAME=/DRIVE_03_012,SAP=0C00,PROFILE=CLT2__CP_L4_INT
ADDRESS14=   192.168.214.2,LINE=10,NAME=/DRIVE_03_013,SAP=0D00,PROFILE=CLT2__CP_L4_INT
ADDRESS15=   192.168.214.2,LINE=10,NAME=/DRIVE_03_014,SAP=0E00,PROFILE=CLT2__CP_L4_INT
ADDRESS16=   192.168.214.2,LINE=10,NAME=/DRIVE_03_015,SAP=0F00,PROFILE=CLT2__CP_L4_INT
name=Machine_1
[param NCU_2]
nck_address= 192.168.214.1,LINE=10,NAME=/NC,SAP=030d,PROFILE=CLT1__CP_L4_INT
plc_address= 192.168.214.1,LINE=10,NAME=/PLC,SAP=0201,PROFILE=CLT1__CP_L4_INT
ADDRESS2=    192.168.214.1,LINE=10,NAME=/CP,SAP=0501,PROFILE=CLT2__CP_L4_INT
ADDRESS10=   192.168.214.1,LINE=10,NAME=/DRIVE_00_000,SAP=0201,SUBNET=0000-
00000000:000,PROFILE=CLT2__CP_L4_INT
ADDRESS11=   192.168.214.1,LINE=10,NAME=/DRIVE_03_003,SAP=0900,PROFILE=CLT2__CP_L4_INT
ADDRESS12=   192.168.214.1,LINE=10,NAME=/DRIVE_03_011,SAP=0B00,PROFILE=CLT2__CP_L4_INT
ADDRESS13=   192.168.214.1,LINE=10,NAME=/DRIVE_03_012,SAP=0C00,PROFILE=CLT2__CP_L4_INT
ADDRESS14=   192.168.214.1,LINE=10,NAME=/DRIVE_03_013,SAP=0D00,PROFILE=CLT2__CP_L4_INT
ADDRESS15=   192.168.214.1,LINE=10,NAME=/DRIVE_03_014,SAP=0E00,PROFILE=CLT2__CP_L4_INT
ADDRESS16=   192.168.214.1,LINE=10,NAME=/DRIVE_03_015,SAP=0F00,PROFILE=CLT2__CP_L4_INT
name=Machine_2
```

References

The "Safety Integrated" function is described in detail in:

Function Manual for Safety Integrated

2.2.1.2 Link module

The NCU-link communication takes place via link modules A link module is an optional PROFINET module for isochronous real-time communication (IRT) via Ethernet. The link module can be used only for the NCU-link communication. It is not possible to use a link module for general PROFINET communication.

The option slot is required on the NCU module for the link module.

Note

There is only one option slot on the NCU module. This prevents the parallel use of an NCU-link module and other optional modules.

Link module and NCU modules

The Communication Board Ethernet CBE30-2 is available as link module for the "NCU710.3 PN", "NCU720.3 PN" and "NCU730.3 PN" NCU modules. The option slot is required on the NCU module for the link module.

References

Manual NCU7x0.3 PN; "Connectable components" > "CBE30-2" section

2.2.1.3 Parameter assignment: NC system cycles

As basic requirement for link communication, the following system cycle clocks must be set the same in **all** NCUs involved in the NCU group:

- Basic system cycle clock
- Position control cycle clock
- Interpolator cycle clock

Basic system cycle clock

The DP cycle set for isochronous communication in the STEP 7 project is used as basic system cycle clock. The current basic system cycle clock is displayed in machine data:

MD10050 $MN_SYSCLOCK_CYCLE_TIME

Note

Manual alignment across several communications buses

If several isochronous communication buses (PROFIBUS 1 ... n, PROFINET 1 ... m) have been configured on an NCU, the same DP cycle time in STEP 7 HW Config must be set for each communication bus.

Depending on the position control cycle clock

For the SINUMERIK 840D sl, the ratio between the basic system cycle clock and the position control cycle clocks is fixed (1:1) and cannot be changed. Because only certain position control cycle clocks can be set for the NCU link, only these position control cycle clocks can be set as the basic system cycle clock or DP cycle time. See the next paragraph "Position control cycle clock".

Position control cycle clock

The position control cycle clock is set as a ratio to the basic system cycle clock. For the SINUMERIK 840D sl, the ratio is fixed (1:1) and cannot be changed. The current position control cycle clock is displayed in the machine data:

MD10061 $MN_POSCTRL_CYCLE_TIME

Note

Permitted position control cycle clocks

For the NCU link, depending on the number of NCUs in the link group, only the following position control cycle clocks [ms] are set:

- **2 NCU**: 2.0, 2.5, 3.0, 3.5, 4.0
- **3 NCU**: 3.0, 3.5, 4.0

See Section "Configuration (Page 85)".

Interpolator cycle clock

The interpolator cycle clock is set as a ratio of the basic system cycle clock. The setting is made via the following machine data:

MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO

The actual interpolator cycle clock is displayed in machine data:

MD10071 $MN_IPO_CYCLE_TIME

Notes on setting

Cycle clock settings

It is recommended the following settings are made:

- The default 90% setting for the CPU time share of the NCK should be retained:
  MD10185 $MN_NCK_PCOS_TIME_RATIO
- The system cycle clocks must be set so that the **average** system utilization caused by the interpolator and the position controller does not exceed 60% in normal program operation. As **maximum value**, 90% should be not be exceeded.
  Note:
  In **special circumstances**, maximum values > 100% can be displayed.

Lowest cyclic communication load

The lowest cyclic communication load during the link communication results for a cycle clock ratio of interpolator cycle clock to position control cycle clock of 1:1. With activated "Dynamic Servo Control (DSC)" drive function, this is generally the most effective setting.

Disadvantage: No telegram repetitions are possible for the link communication.

Time frame of the NC/PLC interface update

The following setting is active during the run-up of all NCUs involved in the link group:

MD18000 $MN_VDI_UPDATE_IN_ONE_IPO_CYCLE = 1

This causes the NC/PLC interface to be read and written completely in just one interpolation cycle.

2.2.1.4 Parameter assignment: Link communication

NC-specific machine data

Number	Identifier $MN_	Meaning
MD12510	NCU_LINKNO	Unique numerical identification of the NCU within the link group. The identifiers must be assigned without any gaps in ascending ordering starting from 1. Value range: 1, 2, ... Maximum NCU number **Note:** The NCU to which the value **1** is assigned as NCU identification is the **master** NCU of the link group. The parameterization of link axes and axis containers may only be made with the machine and setting data of the master NCU.
MD18780	MM_NCU_LINK_MASK.Bit 0	Activation of the link communication
MD18781	NCU_LINK_CONNECTIONS	Number of internal link connections **Note:** It is recommended to retain the default value 0 (determination of the number by the NC).
MD18782	MM_LINK_NUM_OF_MODULES	Number of NCUs to be connected with one another via NCU link.

2.2.1.5 Configuration

The NC system software supplies specific configurations for each supported combination of NCU number and position controller cycle of a link group (see section "Parameter assignment: NC system cycles (Page 83)").

During the system startup, the appropriate configuration will be loaded (as specified by the values parameterized in the machine data):

- MD18782 $MN_MM_LINK_NUM_OF_MODULES (number of NCUs of the line group))
- MD10061 $MN_POSCTRL_CYCLE_TIME (position controller cycle)

Note

For applications, in which the standard configurations that have been supplied cannot be used, please contact your local Siemens sales person.

2.2.1.6 Wiring the NCUs

The numerical sequence of the NCUs within a link group is defined in the NCUs using the following machine data:

MD12510 $MN_NCU_LINKNO = <NCU number>, with NCU number = 1 ... max. NCU number

Cabling

Starting from the NCU1, the NCU link modules should be wired up in the NCU number sequence according to the following schematic: NCU(n), Port 0 → NCU(n+1), Port 1

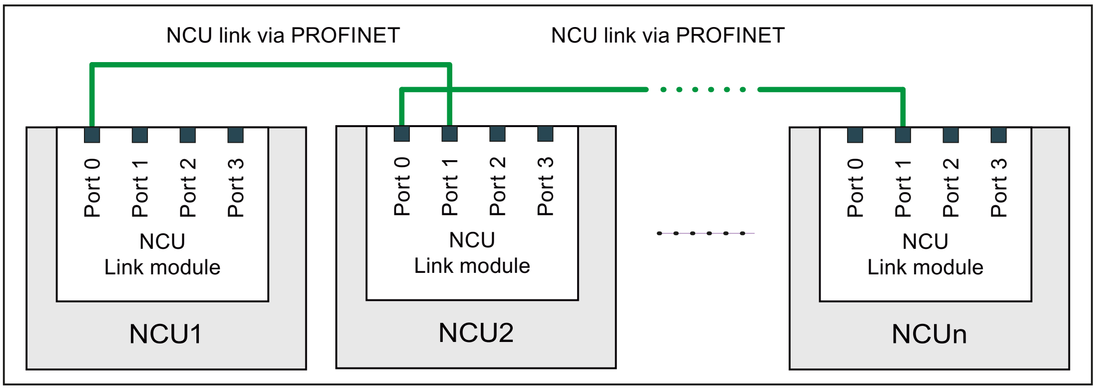


Figure 2-6 Wiring schematic, NCU link

2.2.1.7 Activation

The link communication is activated using the following machine data:

MD18780 $MN_ MM_NCU_LINK_MASK, Bit 0 = 1

Note

Activation time

It is recommended to activate the link communication only after complete commissioning of the entire functionality on all participating NCUs has been done.

2.2.2 Link variables

Complex systems with several NCUs require for the complete system coordination of the manufacturing processes a cyclic exchange of user-specific data between the NCUs. The data exchange is performed via the link communication and a special memory area, the link variables memory available for each NCU.

The size and data structure of the link variables memory can be specified for each specific user. The data stored in the link variables memory is addressed using a special $A_DLx link variable.

These are system-global user variables that can be read and written in part programs and cycles by all NCUs involved in a link group if link communication has been configured. Unlike global user variables (GUD), link variables can also be used in synchronized actions.

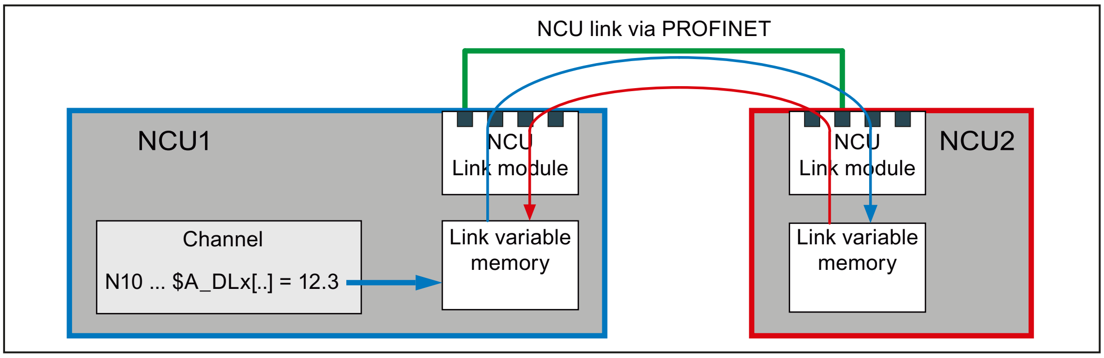


Figure 2-7 Link variables

Requirement

The use of link axes requires a link communication defined in accordance with "Section Link communication (Page 79)".

Link variable as NCU-global user variable

Note

If an NCU Is not a node of a link group or the link variables are not required for communication via NCU link, the link variables can be used as the NCU-global user variables.

2.2.2.1 Properties of the link variables memory

Parameterizing the memory size

The size of the link variables memory in bytes is set with the following machine data:

MD18700 $MN_MM_SIZEOF_LINKVAR_DATA (size of the link variables memory)

The setting for the size of the link variables memory should be identical for all NCUs involved in the link group. If the memory sizes are different, the largest value assigned is used.

Initialization

After an NCU is powered up, the link variables memory is initialized with 0.

Structure

From the point of view of the system, the link variables memory is an unstructured memory area that is available for link communication purposes. The link variables memory is structured by the user/machine manufacturer alone. Depending on how the data structure is defined, the link variables memory is accessed by means of data format-specific link variables.

System-wide alignment

Once a link variables memory has been written to, the changes that have been made to the data are transferred to the link variables memories of all other NCUs involved in the link group. The link variables memories are usually updated by means of link communication within two interpolation cycles.

2.2.2.2 Properties of the link variables

The link variables memory is accessed via the following data format-specific link variables:

Data type 1)	Designation	Data format 2)	Bytes 2)	Index i 3)	Value range
UINT	$A_DLB[i]	BYTE	1	i = n * 1	0 ... 255
INT	$A_DLW[i]	WORD	2	i = n * 2	-32768 ... 32767
INT	$A_DLD[i]	DWORD	4	i = n * 4	-2147483648 ... 2147483647
REAL	$A_DLR[i]	REAL	8	i = n * 8	$\pm(2,2*10^{-308} \ldots 1,8*10^{+308})$

1) Data type of link variables when used in part program/cycle

2) Data format of link variables or number of bytes addressed by the link variables in the link variables memory

3) The following must be noted for index i:

- Index i is a byte index that relates to the beginning of the link variables memory.
- The index must be selected so that the bytes addressed in the link variables memory are located on a data format limit ⇒ index i = n * bytes, where n = 0, 1, 2, etc.
  - `$A_DLB[i]`: i = 0, 1, 2, ...
  - `$A_DLW[i]`: i = 0, 2, 4, ...
  - `$A_DLD[i]`: i = 0, 4, 8, ...
  - `$A_DLR[i]`: i = 0, 8, 16, ...

Write

A link variable is written with main-run synchronism.

Read

A preprocessing stop is initiated when a link variable is read.

Checks

The following checks are performed for the link variables and link variables memory:

- Observance of the value range limits
- Access to format limit
- Observance of defined memory area in link variables memory

The user/machine manufacturer is solely responsible for preventing the following errors:

- Accessing with incorrect data format
- Accessing the wrong address (index i)
- Reciprocal overwriting of the same data item by multiple channels of a single NCU or different NCUs
- Reading a data item before it has been updated by a channel of its own NCU or of a different NCU

Note

Data consistency

The user/machine manufacturer is solely responsible for ensuring data consistency within the link variables memory, both on a local-NCU basis and across NCUs.

2.2.2.3 Write elements

In the case of write access to the link variables memory (e.g. $A_DLB[4] = 21), what is known as a link variables write element is required for managing the write process within the system. The maximum number of write elements that are available for each interpolation cycle is set by means of the following machine data:

MD28160 $MC_MM_NUM_LINKVAR_ELEMENTS

The maximum number of write elements thus restricts the number of link variables that can be written during each interpolation cycle.

2.2.2.4 Dynamic response during write

The link variables are written with main-run synchronism. The new value may be read by the other channels in its own NCU no later than the next interpolation cycle. It can be read in the next part program block in its own channel.

The channels of the other NCUs in the link group normally see the new value after two interpolation cycles. However, the limited bandwidth can lead to delays in transferring write requests to the other NCUs in the link group (message delays). Causes for a message delay can be:

- Writing a large number of link variables in an interpolation cycle
- Writing link variables and a the request for an axis container rotation in the same interpolator cycle
- Writing link variables and transferring an alarm in the same interpolation cycle

2.2.2.5 System variable

NC-specific system variable

Identifier	Meaning
$AN_LINK_TRANS_RATE_LAST	The number of write requests that were still free in the last interpolator cycle.
$AN_LINK_TRANS_RATE_LAST_SUM[<n>]	The number of write requests that were still free in the last interpolator cycle in the send direction to the specified NC <n> (NCU number).
$AN_LINK_CONN_SIZE_LINKVAR	The number of bytes to be transferred for a write request of a link variable.
$AN_LINK_CONN_SND[<n>]	The maximum number of bytes that can be transferred per interpolator cycle from the current NCU to the specified NCU.
$AN_LINK_CONN_RCV[<n>]	The maximum number of bytes that can be transferred per interpolator cycle from the specified NCU to the current NCU.
<n>: NCU number according to MD12510 $MN_NCU_LINKNO of the relevant NCU	

Note for: $AN_LINK_CONN_SIZE_LINKVAR

The writing of a link variable causes the number of bytes specified in $AN_LINK_CONN_SIZE_LINKVAR to be transferred via the not-cyclic link communication. The number is independent of the format of the link variables.

The maximum number of write requests that can be transferred to the specified NCU for each interpolator cycle is calculated as:

Max. number of write requests = $AN_LINK_CONN_SND[<n>] / $AN_LINK_CONN_SIZE_LINKVAR

Channelspecific system variable

Identifier	Meaning
$A_LINK_TRANS_RATE [1)]	Number of write requests that still can be transferred in the current interpolator cycle.
1) Application example, refer to Section: "Synchronization of a write request (Page 91)"	

2.2.2.6 Synchronization of a write request

If certain applications require the new value of a link variable to be transferred to the other NCUs in the link group in precisely two interpolation cycles, writing to the link variable must be made in a synchronized action. In the synchronized action, writing to the link variable is only executed if in the actual interpolator cycle, the write request can still be executed. The $A_LINK_TRANS_RATE system variable contains the number of write requests that can still be transferred in the current interpolator cycle.

In the following example, a link variable, data type WORD (2 bytes) and a link variable, data type DWORD (4 bytes), are written:

Program example

```
N120 WHEN $A_LINK_TRANS_RATE > 0 DO $A_DLW[0] = 9
N125 WHEN $A_LINK_TRANS_RATE > 0 DO $A_DLD[2] = 7
N130 G4 F1
```

The synchronized action in `N120` is performed only when the write request can be transferred in the same interpolator cycle to the other NCU of the link group. In this case, the $A_LINK_TRANS_RATE system variable is also decremented in the same interpolator cycle so that the updated value is available for the synchronized action in the following `N125` block.

2.2.2.7 Example: Structure of the link variables memory

The following data are defined for the link communication:

Data format	Number	Bytes per data	Bytes required
BYTE	2	1	2
WORD	1	2	2
DWORD	3	4	12
REAL	1	8	8
required size of the link variables memory:			**24**

Memory structure

The data is arranged in the link variables memory as follows, with the data format limits taken into account:

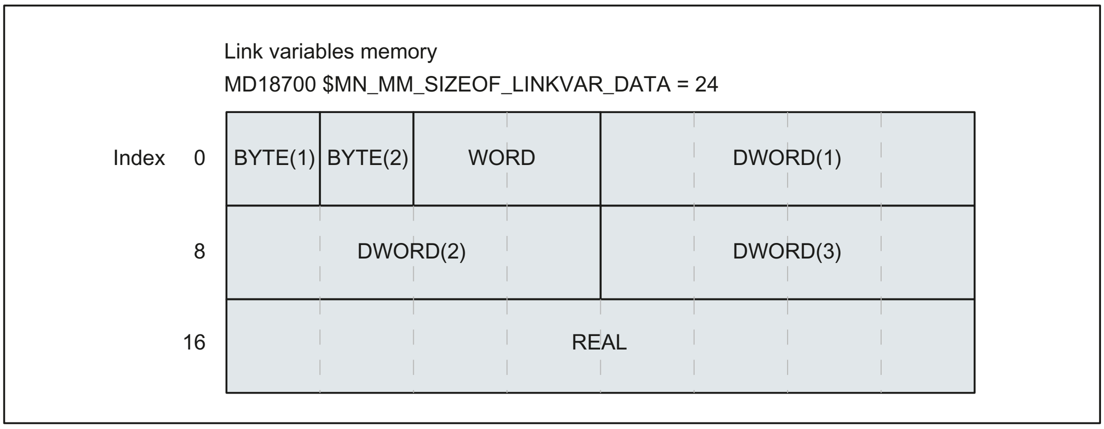


Figure 2-8 Example: Structure of the link variables memory

Note

Memory structure

The data in the link variables memory is always arranged randomly and may therefore appear different (although the data format limits will still be taken into account).

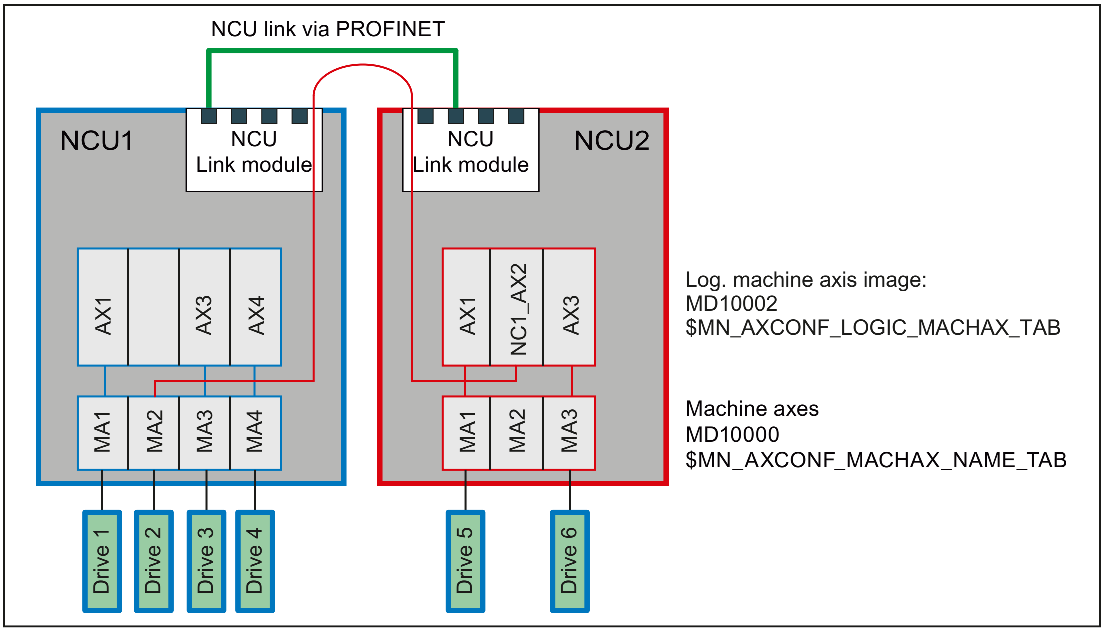
Access via a link variable must be programmed as follows, in accordance with the memory structure defined:

Program code	Description
`$A_DLB[0]`	`; BYTE(1)`
`$A_DLB[1]`	`; BYTE(2)`
`$A_DLW[2]`	`; WORD`
`$A_DLD[4]`	`; DWORD(1)`
`$A_DLD[8]`	`; DWORD(2)`
`$A_DLD[12]`	`; DWORD(3)`
`$A_DLR[16]`	`; REAL`

2.2.2.8 Example: Read drive data

Task

A system contains two NCUs (NCU1/NCU2). The NCUs are connected via the NCU link. The MA2 machine axis of NCU1 (drive 2) travels in interpolation mode as link axis for NCU2. The actual current value of drive 2 should be transferred for evaluation from NCU1 to NCU2. The figure below shows the basic design of the system.



Requirement

The actual current value for drive 2 (NCU1/MA2) can be read via the $VA_CURR system variable. In the case of PROFIdrive-based drives, the following machine data item needs to be set for this purpose:

MD36730 $MA_DRIVE_SIGNAL_TRACKING = 1 (acquisition of additional actual drive values)

Setting the machine data makes the following drive actual values available:

- $AA_LOAD, $VA_LOAD (drive capacity utilization in %)
- $AA_POWER, $VA_POWER (drive active power in W)
- $AA_TORQUE, $VA_TORQUE (drive torque setpoint in Nm)
- $AA_CURR, $VA_CURR (actual current value of axis or spindle in A)

Programming

NCU1

A static synchronized action is used to write cyclically in the interpolation cycle the actual current value $VA_CURR of axis 2 (NCU1/MA2) via the link variable $A_DLR[0] (REAL value) to the first 8 bytes of the link variables memory:

Program code
`IDS=1 WHENEVER TRUE DO $A_DLR[0]=$VA_CURR[MA2]`

NCU2

With a static synchronized action, the transferred actual current value is read cyclically in the interpolator cycle via link variable $A_DLR[0]. If the actual current value is greater than 23 A, alarm 61000 is displayed.

Program code
`IDS=1 WHEN $A_DLR[0] > 23.0 DO SETAL(61000)`

2.2.3 Link axes

2.2.3.1 General information

A link axis is designated as a machine axis for which the setpoints are generated by a different NCU than that to which the machine axis is physically connected. Link axes in conjunction with axis containers (see Section "Axis container (Page 100)") permit in complex plants, such as rotary indexing machines with several NCUs, the alternate use of machine axes of the NCU of the link group.

The following figure shows as example an MA1 machine axis connected to the NCU1. The MA2 machine axis is connected to the NCU2. The X and Z channel axes travel in interpolation mode by a part program that runs in a channel of NCU1. The setpoints are generated in the NCU1 interpolator. For machine axis MA1, they are forwarded to the position control of the NCU1. For machine axis MA2, they are transferred via the NCU link to the position control of the NCU2 and output there to the drive.

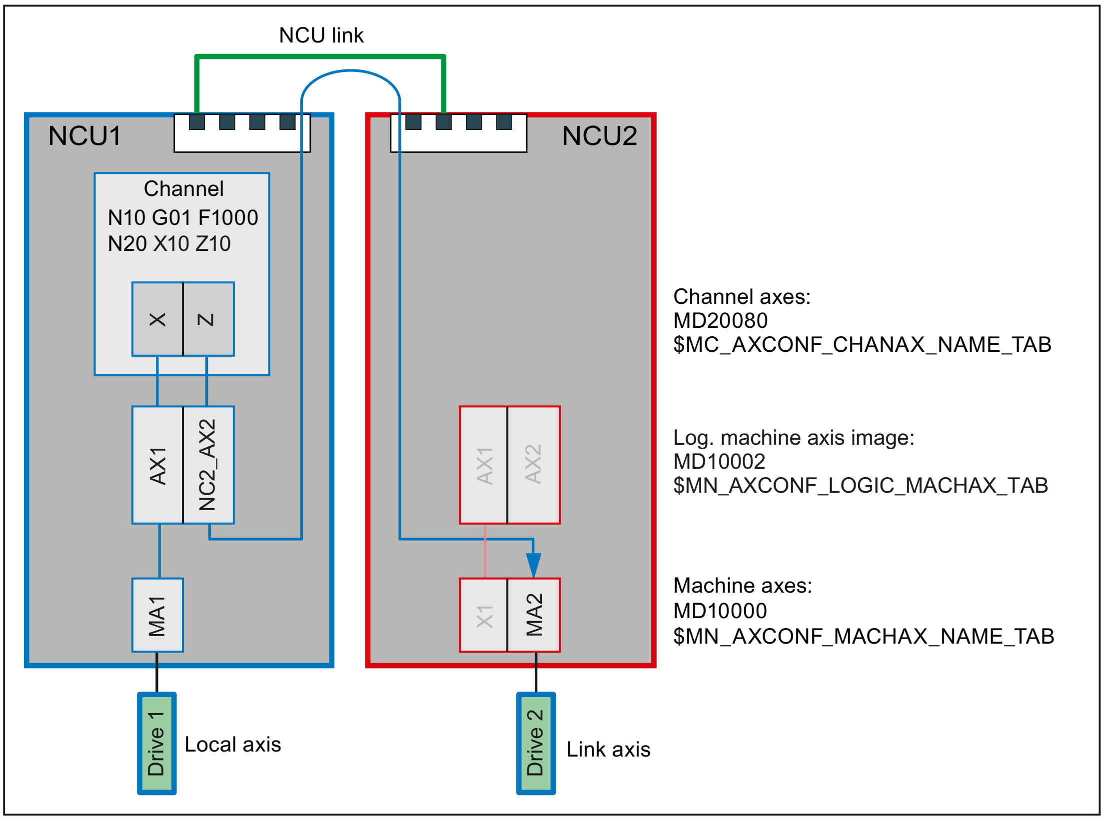


Figure 2-9 Link axes

Requirement

The use of link axes requires a link communication defined in accordance with "Section Link communication (Page 79)".

Home NCU

The home NCU of a link axis is the NCU on which it is physically connected as machine axis. The position control and the exchange of axial NC/PLC interface signals of a link axis always occurs on the home NCU. The generation of the setpoint can in principle be performed on any NCU of the link group.

In the above figure:

- NCU1: Home NCU of machine axis MA1
- NCU2: Home NCU of machine axis MA2

2.2.3.2 Name of a link axis

The name of a link axis is composed of the identifier for the home NCU on which the machine axis is physically connected and the general machine axis name AXn:
NC<ID>_<axis>

<ID>:	Identification of the NCU of the link group in accordance with: MD12510 $MN_NCU_LINKNO See Section "Parameter assignment: Link communication (Page 85)"
<axis>:	General machine axis name: AX1, AX2, AX3, ...

2.2.3.3 Parameterization

Assignment: Geometry axis or special axis to link axis

Direct assignment

A geometry or special axis can be assigned directly to a link axis in the logical machine axis image by specification of the link axis name:

MD10002 $MA_AXCONF_LOGIC_MACHAX_TAB[<axis>] = <link axis>

Parameters	Meaning
<axis>:	Machine axis index: 0, 1, 2, … (max. number of machine axes - 1)
<link axis>:	Name of the link axis: NCx_AXy, see Section "Name of a link axis (Page 96)"

Indirect assignment

A geometry or special axis can be assigned to a link axis indirectly in the logical machine axis image by specification of a container slot. The container slot then contains the actual name of the link axis, as described above. This is called a container-link axis. See also Section: "Axis container (Page 100)".

Synchronous setpoint output

A delay of one interpolator cycle results during the transfer of the setpoints of a link axis for the setpoint-generating NCU to the home NCU. To ensure that the setpoints for the interpolation of local axes and link axes are output to the drives at exactly the same time, this delay must be compensated. To do this, on the setpoint-generating NCU, the number of buffer elements of the buffer between the interpolator and the position controller must be set one element larger than the number of buffer elements of the home NCU:

MD18720 $MN_MM_SERVO_FIFO_SIZE= 3

Example

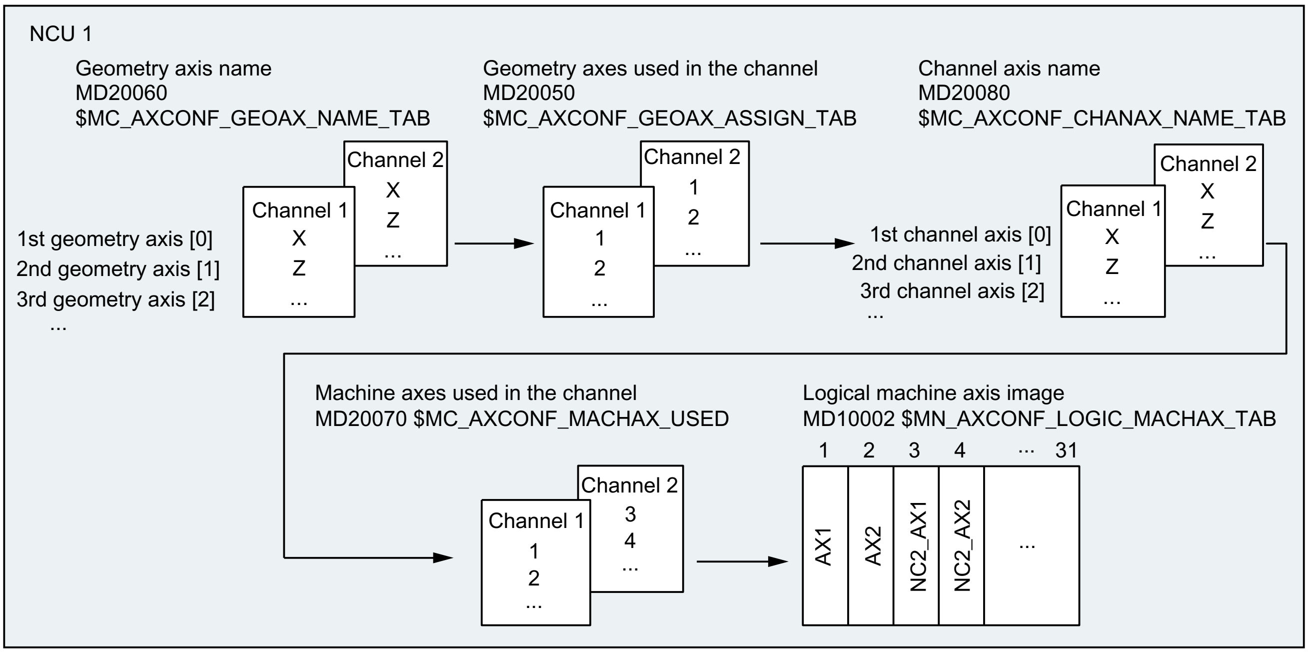


Figure 2-10 Example: Parameterization of link axes

Channel 1

The local AX1/AX2 machine axes of the NCU1 are assigned to the X/Z geometry axes.

Channel 2

The NC2_AX1/NC2_AX2 link axes of the NCU2 are assigned to the X/Z geometry axes.

2.2.3.4 Auxiliary function output for spindles

During program execution and after block search with "search via program test" (SERUPRO), the pre-defined auxiliary functions S, M3, M4, M5, M19 and M70 are output channel-specific on the NCU on which the spindle has been programmed and also axis-specific on the home NCU.

Output of the channel-specific auxiliary functions

- DB21, ... DBW68 (extended address of the M function)
- DB21, ... DBD70 (M function 1)
- DB21, ... DBW98 (extended address of S function 1)
- DB21, ... DBD100 (S function 1)

Output of the axis-specific auxiliary functions

- DB31, ... DBD78 (F function for axis)
- DB31, ... DBW86 (M function for spindle)
- DB31, ... DBD88 (S function for spindle)

References

Detailed information about the auxiliary function output can be found in:
Function Manual, Basic Functions, "Help function outputs to the PLC (H2)" section

2.2.3.5 Supplementary conditions

Maximum number of machine axes

Even for the use of link axes, as previously, the maximum number of concurrently-usable geometry and special axes as well as machine axes are still available for each NCU type.

"Lead link axes and "link axes" functions

As the functions "lead link axes" and "link axes" require different settings in machine data MD18720 $MN_MM_SERVO_FIFO_SIZE, they cannot be used within a link group at the same time.

Alarms: General behavior

If an error is detected at the position control level of the home NCU of a link axis and the relevant alarm does **not** have "NC not ready" as response, the alarm is transferred via the NCU link to the setpoint-generating NCU and then output.

Alarms: Behavior for emergency stop

If an emergency stop request is activated for an NCU via the NC/PLC interface, all axes physically connected to this NCU will be switched to the "follow-up" mode. This also affects link axes whose setpoints are currently generated by other NCUs. Under the assumption that no further practical machining operations are then possible on these NCUs, an additional alarm will be generated that stops all dependent axis movements.

Alarm acknowledgement

This additional alarm must be acknowledged by an NC reset. If the original alarm is still active at this time, although the additional alarm can be acknowledged, an additional self-clearing alarm will be generated that prevents a traversal of the axes or a program start until the original alarm has been acknowledged.

Alarms: Response for "NC not ready" alarm response

If an error is detected at the position control level of the home NCU of a link axis and the relevant alarm does not have "NC not ready" as response, the alarm is transferred to the setpoint-generating NCU via the NCU link and output there. The alarm output is also made on the home NCU.

Under the assumption that no further practical machining operations are then possible on the setpoint-generating NCU, an additional alarm will be generated that stops all dependent axis movements.

Alarm acknowledgement

See alarm acknowledgement under "Alarms: Behavior for emergency stop"

Alarms: Behavior for "Mode group not ready" alarm response

If an error is detected within a mode group with several channels and the relevant alarm has "Mode group not ready" as response, this causes the traversing movements in all channels of the mode group to stop. If the traversing movements are generally independent of each other, the response via the following machine data item can be changed to "Channel not ready":

MD11412 $MN_ALARM_REACTION_CHAN_NOREADY = TRUE

Effect

In the NC/PLC interface, instead of signal DB11 DBX26.3 (mode group ready), the signal DB21, DBX36.5 (channel ready), is reset.

Advantage

The alarm response remains limited to the channel in which the error is detected. If required, the PLC user program can initiate further responses.

Requirement

There is no higher-priority alarm response than "Mode group not ready".

Compensations

The following compensations are **not** available:

- Link axes: Quadrant error compensation (QEC)
- Container link axes: Sag compensation (CEC)

Switch-off an NCU in a link group

If an NCU in a link group is switched off, the machining on all other NCUs of the link group will be cancelled and an alarm issued.

Powering up an NCU group

If an NCK reset is triggered on an NCU in a link group, it will also be transferred to all other NCUs of the link group so that all NCUs of the link group perform a warm restart.

Nibbling and punching technologies

The fast inputs/outputs required for the nibbling and punching must be connected and parameterized on the NCU on which the part program is processed and the axes interpolated. The commands for "High-speed nibbling and punching", e.g. `PONS` and `SONS`, are not available for link axes.

Frames

A link axis is permitted in a Frame command only when it is a geometry axis. The Frame command changes only the geometry in the channel to which the link axis is currently assigned.

Speed/torque coupling, master-slave

The drives of all axes/spindles of a master-slave group must be connected to the same NCU. The master axis can, however, be traversed as link axis for the channel of another NCU.

2.2.4 Axis container

2.2.4.1 General information

An axis container is a circular data structure with a parameterized number of elements. These elements together with axis containers, are designated as slots (Slot 1, Slot 2, ... Slot n). The slots allow a variable assignment of geometry and/or special axes to machine axes. The entry in a slot can reference an NCU-local machine axis (container axis) or a link axis (container-link axis).

The following figure shows an axis container with four slots. The container axes refer in the current position of the axis container to the following machine axes:

Container axis	Machine axis
CT1_SL1	NCU 1: AX1
CT1_SL2	NCU 1: AX2
CT1_SL3	NCU 2: AX1
CT1_SL4	NCU 2: AX2

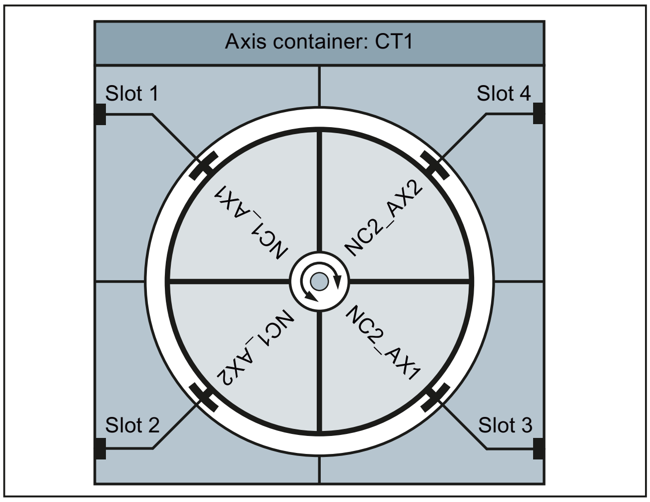


Figure 2-11 Example: Axis container CT1 with four slots

Rules

The following rules must be observed with regard to axis containers:

- All machine axes of an axis container may be assigned to just one channel axis at any one time.
- Multiple slots of an axis container must not reference the same machine axis.
- At any one time, only one channel is authorized to write to a machine axis, directly or via a container axis.
- Several geometry and/or special axes of a channel can also be assigned to the container axes of an axis container.

Assignment: Geometry or special axis → container axis

In the logical machine axis image MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB, a container axis can be assigned as a machine axis via container and slot to a geometry or special axis, e.g. CONTAINER "CT1", slot "1":

MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[n] = CT1_SL1

During the traversing of the geometry or special axis, the machine axis assigned at this time to slot 1 then traverses.

Axis container rotation

An axis container rotation is always performed when all channels involved on the axis container have granted their enable with a program command. After the axis container rotation, other machine axes are assigned to the geometry or special axes of the channels.

The setting data specifies the increment of an axis container rotation.

See Section "Parameterization (Page 103)".

Axis container name

The following program commands can be addressed via the axis container name (<axis container>):

- Program commands:
  - `AXCTSWE(<axis container>)`
  - `AXCTSWED(<axis container>)`
  - `AXCTSWEC(<axis container>)`

The following names are possible:

`CT<container number>`:	The number of the axis container is attached to the CT letter combination. Example: `CT3`
`<container name>`:	Individual name of the axis container set using MD12750 $MN_AXCT_NAME_TAB Example: `A_CONT3`
`<axis name>`:	Axis name of a container axis which is known in the channel involved.

Implicit wait

There is an implicit wait for the completion of a requested axis container rotation if one of the following events has occurred:

- Part program language commands which will cause a container axis assigned to this axis container in this channel to move
- `GET(<channel axis name>)` on a corresponding container axis
- The next `AXCTSWE(<axis container>)` for this axis container

> **Note**
>
> Even an `IC(0)` will result in a wait including synchronization where necessary (block-by-block change in addressing according to incremental dimension even though an absolute dimension has been set globally).

Synchronization with axis position

If the new container axis assigned to the channel after a container rotation does not have the same absolute machine position as the previous axis, then the container is synchronized with the new position (internal REORG).

Note

SD41700 $SN_AXCT_SWWIDTH[<axis container>] is only updated for a new configuration. If, after incremental rotation of the RTM/MS, the position of a circuit before the initial position is reached, the container can be rotated normally in the **forward** direction to reach the initial position of the container again. The drum or rotary table must however be turned **back** to the original position, so that measuring and supply cables are not twisted.

See also

System variable (Page 111)

Programming (Page 110)

2.2.4.2 Parameterization

Machine data

NC-specific machine data

Number	Identifier $MN_	Meaning
MD12750	AXCT_NAME_TAB	Name of the axis container
MD12760	AXCT_FUNCTION_MASK.Bit x	Axis container-specific functions
MD1270x	AXCT_AXCONF_ASSIGN_TABx	Assignment of machine axes to the slots of an axis container
MD18720	MM_SERVO_FIFO_SIZE	Size of the IPO/SERVO data buffer **Note:** The value 3 must be set on all participating NCUs for cross-NCU axis containers.

Name of the axis container

MD12750 $MN_AXCT_NAME_TAB[<index>] = "<name>"

Parameters	Meaning
<index>:	0, 1, … max. axis container index
<name>:	Name of the axis container (e.g. CT1)

Axis container-specific functions

MD12760 $MN_AXCT_FUNCTION_MASK.Bit x = <value>

Parameters	<value>	Meaning
Bit 0:	0	For a direct axis container connection (AXCTSWED), all other channels must be in the RESET state.
	1	For a direct axis container connection (AXCTSWED), only other channels that have the interpolation authorization to axes of the axis container need to be in the RESET state.

The machine data is used to activate the axis container-specific functions.

Assignment of machine axes to the slots of an axis container

MD1270**x** $MN_AXCT_AXCONF_ASSIGN_TAB**x**[<index>] = <axis>

Parameters	Meaning
x:	1 … max. number of axis containers
<index>:	0, 1, … max. slot index
<axis>:	Machine axis name of a local machine axis (e.g. AX1)
	Name of a link axis. See Section "General information (Page 94)".

The slots within an axis container must be assigned in ascending order without gaps, starting with slot index 0.

Setting data

Increment of an axis container rotation

SD41700 $SN_AXCT_SWWIDTH[<index>]=<increment>

Parameters	Meaning
<index>:	0, 1, … max. axis container index
<increment>:	Number of slots through which the axis container rotates

Illustration of the axis container rotation

The axis container rotation is enabled by means of program commands. See Section "Programming (Page 110)".

In the following Fig. 1 (left-hand side), the NCU1_AX1 link axis is assigned as example to slot 1 in the **Initial setting** axis container.

After the rotation with increment of 1 (Fig. 1, right-hand side), the NCU2_ AX2 link axis is assigned to slot 1.

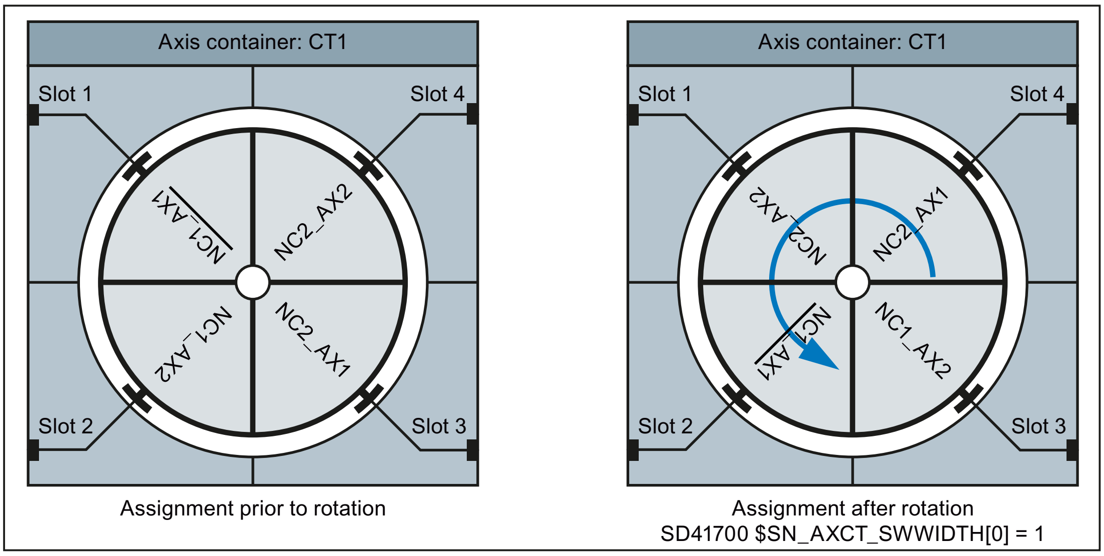


Figure 2-12 Axis container rotation, Fig. 1

Starting with the **initial setting**, after the rotation with increment 2 (Fig. 2, left-hand side), the NCU2_ AX1 link axis is assigned to slot 1.

Starting with the **initial setting**, after the rotation with increment **-1** (Fig. 2, right-hand side), the NCU1_AX2 link axis is assigned to slot 1.

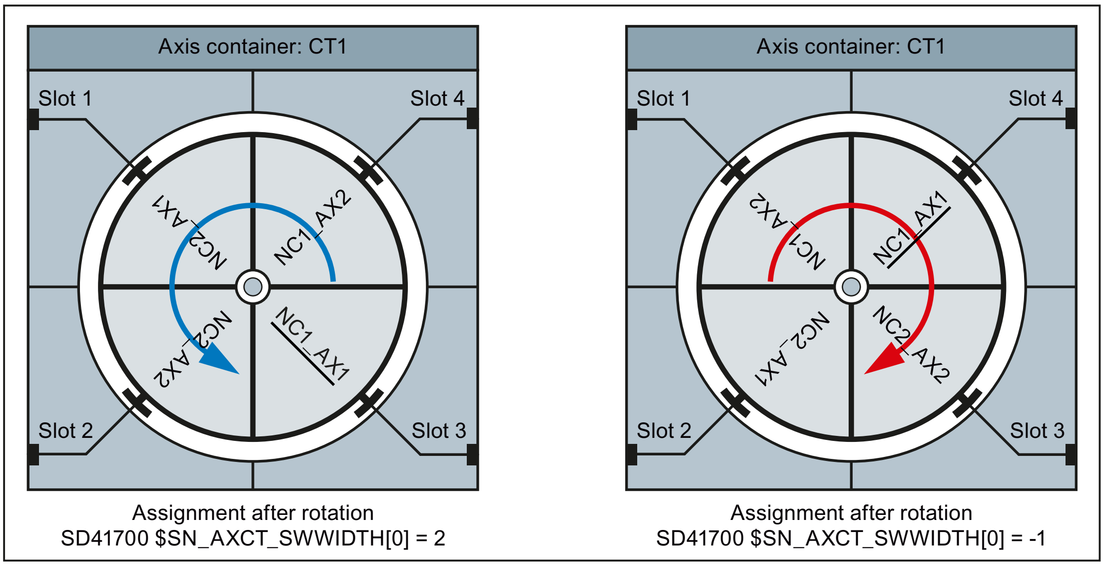


Figure 2-13 Axis container rotation, Fig. 2

Axis container with container-link axes

The parameterization of an axis container that contains container-link axes must be made on the master NCU of the link group (MD12510 $MN_NCU_LINKNO == 1).

Alignment of axial machine data

For container axes, all axial machine data marked with the "CTEQ" (container equal) attribute must have the same value for all container axes. Any different values will be aligned automatically.

Control startup

During the control startup, all machine data will be aligned to the values of the container axis of the first slot. If the value of a machine data item changes, the following message will be displayed: "The axial machine data of the axes in axis container <n> has been adapted".

Activate machine data

If a machine data item of any container axis is changed, the new value also becomes immediately available in all other container axes. The following message is displayed: "Caution: This MD will be set for all container axes".

Slot change

If a slot of an axis container is assigned another machine axis, (MD127xx AXCT_AXCONF_ASSIGN_TAB<x>), the following message is displayed: "The machine data of the axes in axis container <n> will be adapted at the next startup".

Note

Container-link axes

For container-link axes, a machine data alignment is performed for all NCUs of the link group involved on the axis container.

Parameter example

Assumptions

NCU	Components
NCU1:	Channel 1, X/Z geometry axes → 1st/2nd channel axis Channel 2, X/Z geometry axes → 1st/2nd channel axis Machine axes: AX1, AX2 CT1 axis container with four slots
NCU2:	Machine axes: AX1, AX2

Parameter assignment: NCU1

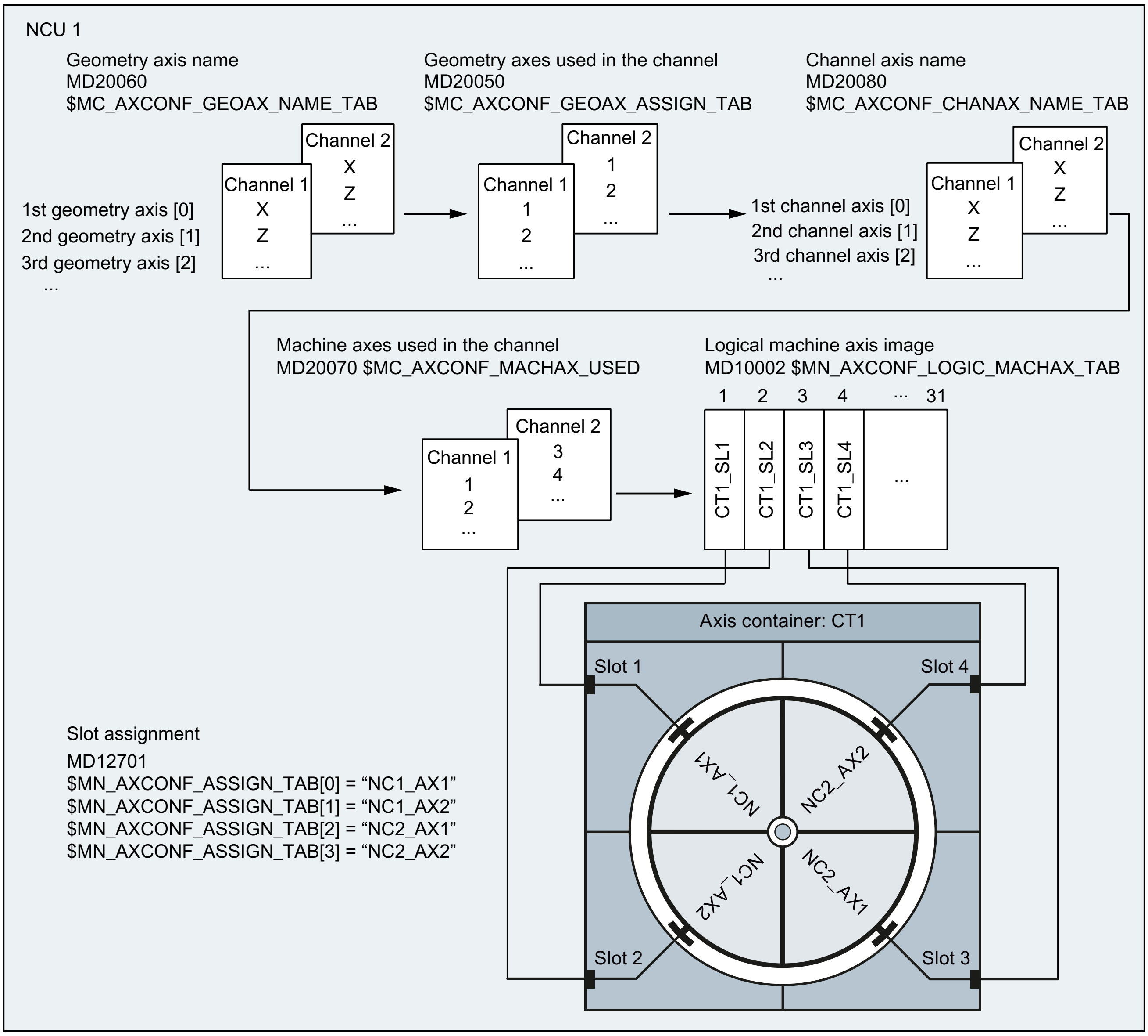


Figure 2-14 Example: Parameter assignment of channel axes and axis containers

Effect

By programming the X and Z geometry axes in the 1st and 2nd channel of the NCU1, the following axes traverse in the current position of the container:

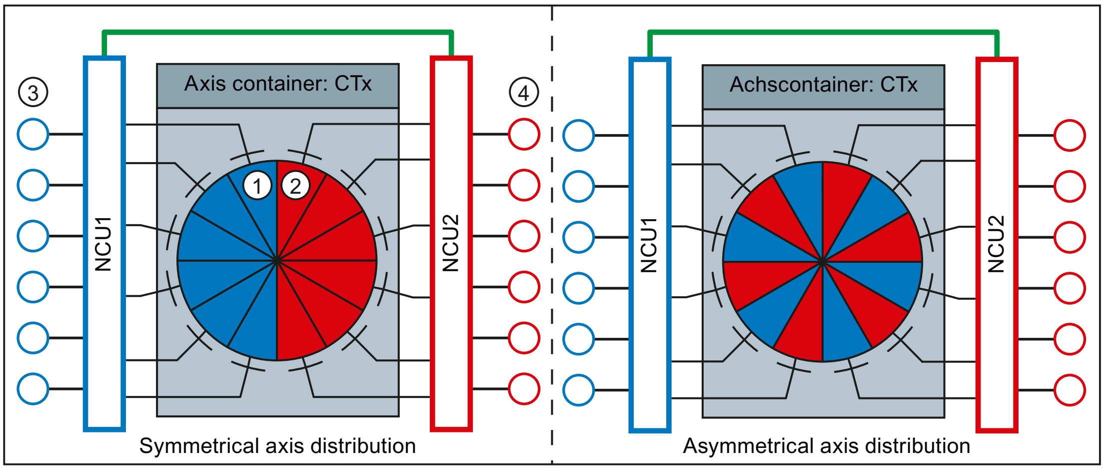
- The local AX1 and AX2 machine axes of the NCU1.
- As container-link axes, the AX1 and AX2 machine axes of the NCU2.

For the axis container rotation, see Section "Programming (Page 110)".

Notes on the parameter assignment

Container axis distribution and communications utilization

In the case of a plant with several NCUs that traverse alternately axes of other NCUs (link axes) in conjunction with axis containers, the type and manner how the link axes are distributed within the axis container decide on the utilization of the link communication.



① Blue slot: Refers to a drive connected to the NCU1
② Red slot: Refers to a drive connected to the NCU2
③ Drives connected to NCU1
④ Drives connected to NCU2

Figure 2-15 Axis distribution

- Symmetrical axis distribution

  In the case of a symmetrical axis distribution, each NCU first traverses only local axes. This means no link communication occurs. Provided all NCUs only traverse link axes, each transition of the axis container increases the utilization of the link communication up to a maximum.

- Asymmetrical axis distribution

  For an asymmetric axis distribution, each NCU traverses local and link axes from the beginning. Unlike the symmetrical axis distribution, this results in a "constant" average utilization of the link communication.

Drive distribution and communications utilization

In a system with several NCUs that in conjunction with axis containers alternately traverse axes of another NCU (link axes), the distribution of the drives connected to the NCU decides the utilization of the link communication.

- Symmetric drive distribution

  In the case of a symmetric drive distribution, the drives addressed via the axis container are connected to both NCUs. On account of this arrangement, the maximum possible number of drives can still be addressed via the logical machine axes images (LAI) on both NCUs.

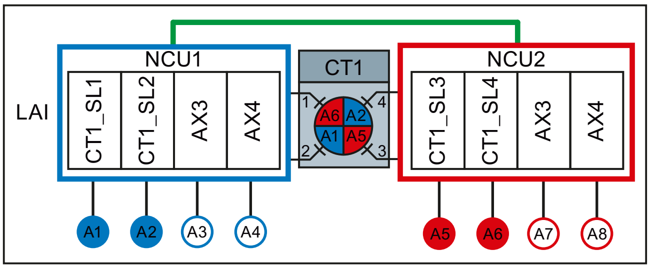


Figure 2-16 Symmetric drive distribution

- Asymmetric drive distribution

  In the case of an asymmetric drive distribution, the drives addressed via the axis container are only connected to NCU1. On account of this arrangement, the maximum possible number of drives can only be addressed via the logical machine axes images (LAI) on NCU2. Only the maximum number minus the drives used by NCU2 can be addressed via the LAI of NCU1. In order to also be able to use the maximum number of drives on NCU1, they must be connected to NCU2 and addressed from NCU1 via the NCU link. This results in a higher cyclic link communication load.

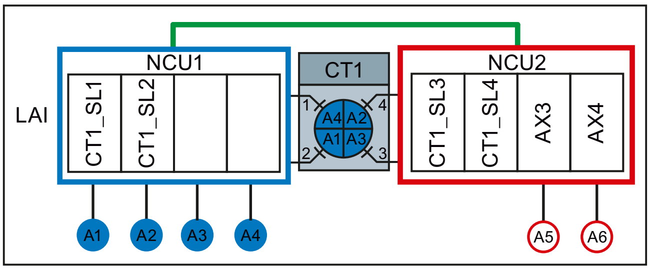


Figure 2-17 Asymmetric drive distribution

2.2.4.3 Programming

Function

The AXCTSWE or AXCTSWED commands are used to enable the rotation of the specified axis container.

Any previously set enable for axis container rotation is cancelled with the AXCTSWEC command.

Syntax

```
AXCTSWE(<ID>)
AXCTSWED(<ID>)
AXCTSWEC(<ID)
```

Meaning

AXCTSWE:	Enable for rotation of the axis container The program processing is not stopped by AXCTSWE. The rotation is performed as soon as all channels involved on the axis container have been enabled. The increment of an axis container rotation is set via the SD41700 $SN_AXCT_SWWIDTH setting data (see Section "Parameterization (Page 103)")
AXCTSWED:	Enable to rotate the axis container without consideration of the other channels involved on the axis container The increment of an axis container rotation is set via the SD41700 $SN_AXCT_SWWIDTH setting data (see Section "Parameterization (Page 103)") **Note** • Command variant to simplify the commissioning of the part program or synchronized action. • The behavior with regard to the other channels involved on the axis container can be specified via: MD12760 $MN_ AXCT_FUNCTION_MASK, bit 0 See Section "Parameterization (Page 103)".
AXCTSWEC:	Canceling the enable to rotate the axis container **Note** The enable for rotating an axis container can only be cancelled when the rotation has yet not been started: $AN_AXCTSWA[<axis container>] == 0 See Section "System variable (Page 111)"

`<ID>`:	Name of the axis container or a container axis:	
	`CT<number>`:	Default identifier of an axis container: MD12750 $MN_AXCT_NAME_TAB Example: `CT1`
	`<Container>`:	User-specific name of an axis container: MD12750 $MN_AXCT_NAME_TAB Example: `CONTAINER_1`
	`<axis>`:	Name of a known container axis in the channel

References

The use of the `AXCTSWEC` command in synchronized actions is described in detail in:

Synchronized Actions Function Manual, Section "Detailed description" > "Actions in synchronized actions" > "Cancel release for axis container rotation (AXCTSWEC)"

2.2.4.4 System variable

Container-specific system variable

System variable	Description
$AC_AXCTSWA[<ID>]	Channel-specific status of the axis container rotation
$AN_AXCTSWA[<ID>]	NCU-specific status of the axis container rotation
$AN_AXCTSWE[<ID>]	Slot-specific status of the axis container rotation
$AN_AXCTAS[<ID>]	Number of slots through which the axis container was just switched through.
ID: Axis container name or name of a container axis	

NC-specific system variable

System variable	Description
$AN_LAI_AX_IS_AXCTAX 1)	Status: LAI axis == container axis of the machine axes in the logical machine axis image (MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB) regarding axis container.
$AN_LAI_AX_IS_LEADLINKAX 1)	Status: LAI axis == lead-link axis
$AN_LAI_AX_IS_LINKAX 1)	Status: LAI axis == link axis
$AN_LAI_AX_TO_IPO_NC_CHANAX[<ID>]	Channel and channel axis number or NCU-ID and the global axis number
$AN_LAI_AX_TO_MACHAX[<ID>]	NCU-ID and axis number of the machine axis
1) Bit mask: Bit **n** ≙ LAI axis (**n**+1) from MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB ID: LAI axis number NCU-ID: Value from MD12510 $MN_NCU_LINKNO	

References

A detailed description of the system variables can be found in:

System Variables, List Manual

See also

Evaluating axis container system variables (Page 124)

2.2.4.5 Machining with axis container (schematic)

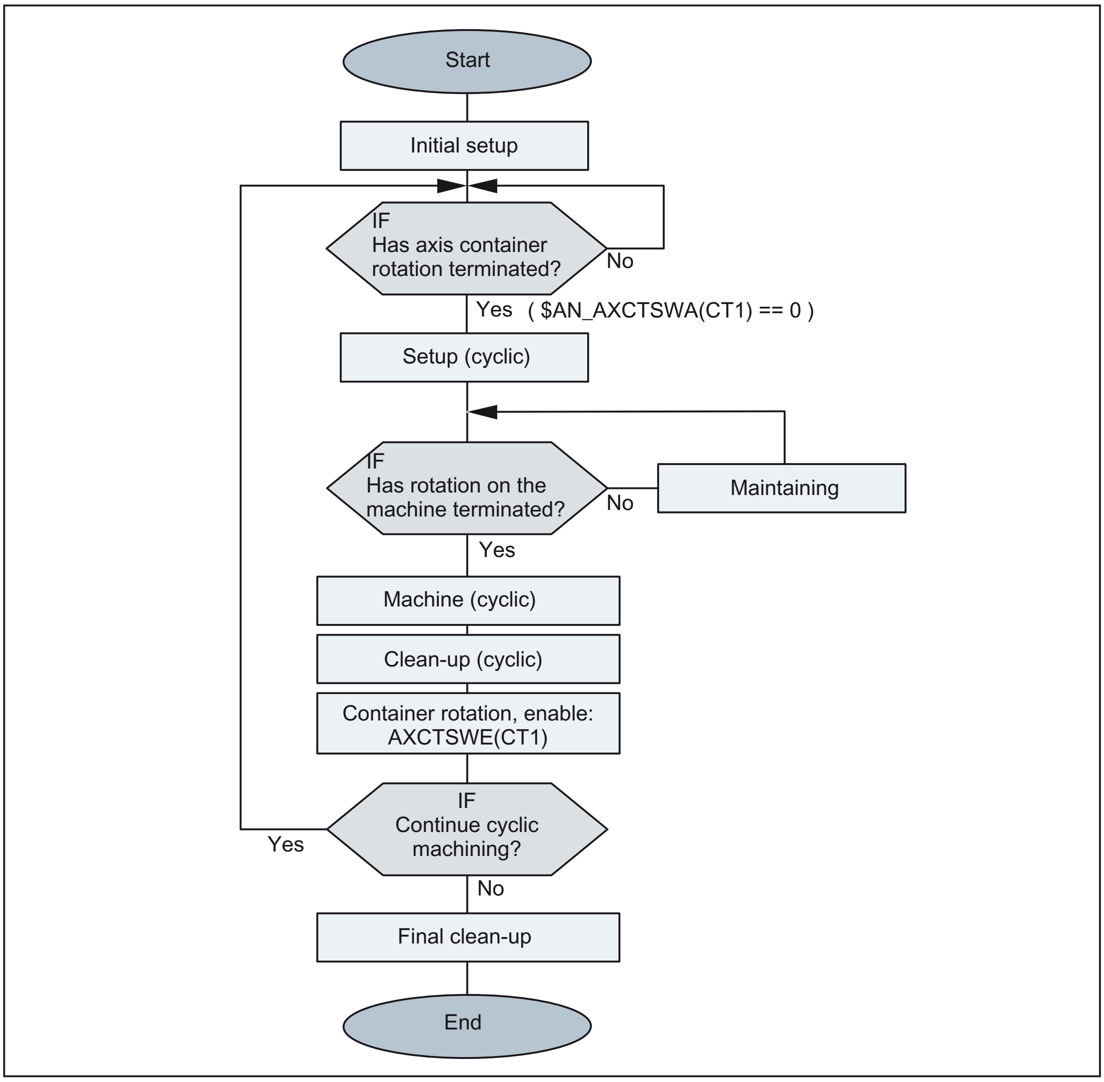


Figure 2-18 Example: Schematic machining sequence for a station of a rotary cycle machine

2.2.4.6 Behavior in different operating states

Startup (Power On)

In the startup the controller, with regard to the slot assignment, the initial state defined in the machine data is always assumed irrespective in which state of the axis container the control was switched off:

MD1270x $MN_AXCT_AXCONF_ASSIGN_TABx

Note

Alignment between setpoint and actual status

After a controller startup, it is the sole responsibility of the user / machine manufacturer to detect any difference between the status of the axis container and the machine status and to compensate for this with a suitable axis container rotation.

Mode change

A container axis whose axis container in the Automatic mode was enabled for rotation cannot be traversed after a change in the JOG mode.

Channel-specific Reset state

As soon as a channel involved on the axis container is in the Reset state, no enable for axis container rotation is required for this channel. It suffices to enable the remaining active channels.

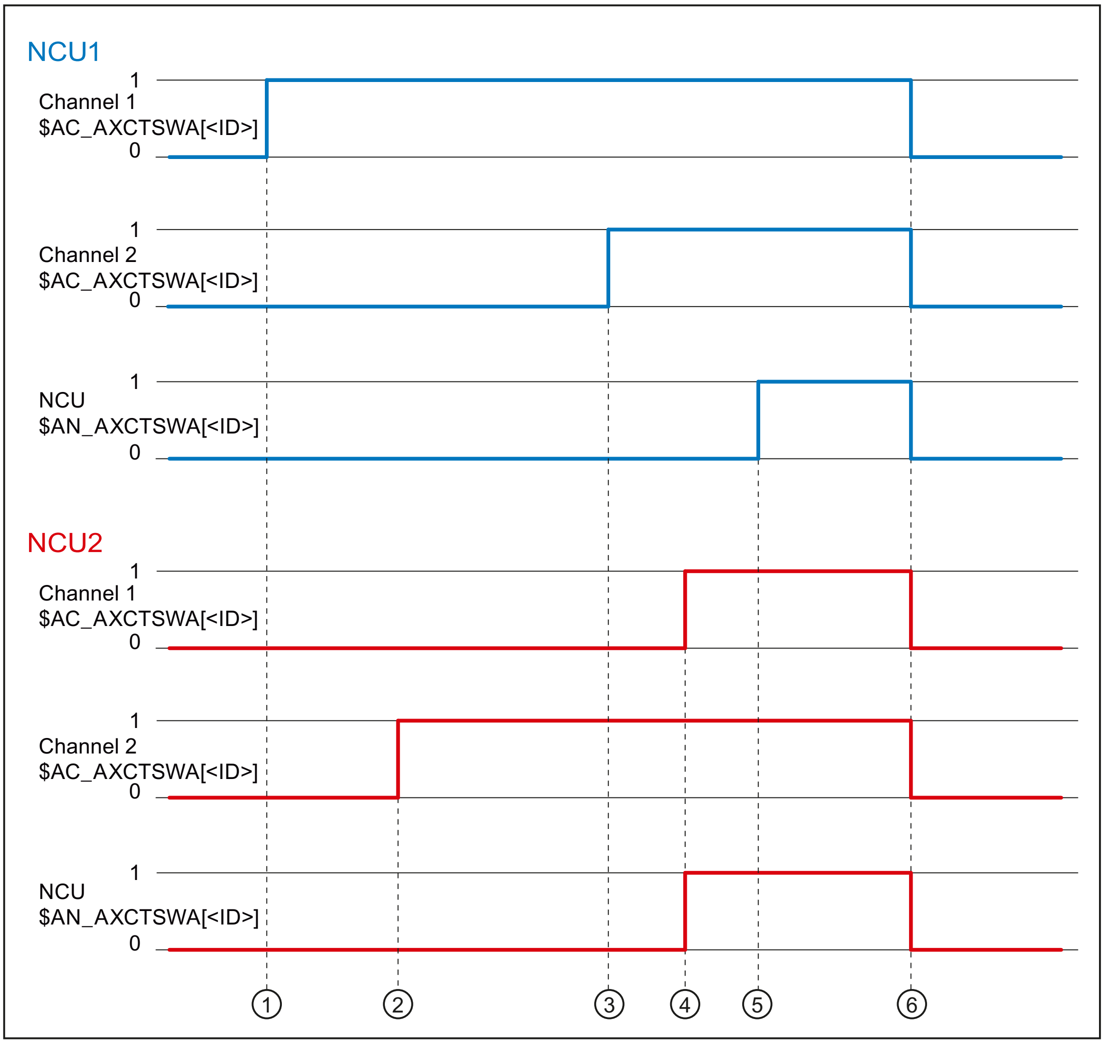
Block search

An axis container rotation (`AXCTSWE`) cannot be enabled and activated in one block, but the enabling and activation commands must be programmed in separate action blocks. In other words, the axis container status changes in response to each separate rotation command as a function of the status of other channels.

2.2.4.7 Behavior when withdrawing the release for axis container rotation

The`AXCTSWE` command enables the channel-specific axis container rotation for an axis container. The `AXCTSWEC` command cancels the enable.

The following figure shows an example the sequence of an axis container rotation as represented in the axis container-specific system variables. Two channels for two NCUs are always involved on the axis container.



① NCU1, channel1: Enable issued using the `AXCTSWE` command

② NCU2, channel2: Enable issued using the `AXCTSWE` command

③ NCU1, channel2: Enable issued via `AXCTSWE` command → all enables of all channels are now present in the NCU1 → the overall enable status of NCU1 is transferred to NCU2 via link communication

④ NCU2, channel1: Enable issued via `AXCTSWE` command → all enables of all channels are now present in the NCU2 → the overall enable status of NCU2 is transferred to NCU1 via link communication

All enable signals for all channels (NCU2 and NCU1) are now present in NCU2 → the axis container rotation is performed in NCU2

⑤ NCU1: All enable signals for all channels (NCU1 and NCU2) are now present in NCU1 → the axis container rotation is performed in NCU1

⑥ NCU1/NCU2: axis container rotation has been completed

Figure 2-19 Cross-NCU enable and axis container rotation

To allow a previously granted enable to be canceled, the enable for at least one of the channels (NCU1 or NCU2) involved on the axis container must still be pending at the time of the cancellation. This means the cancel must be made before time ④.

Withdrawal is no longer possible as soon as all enable signals are available from all channels of all NCUs (instant in time ④). In this case, the `AXCTSWEC` command has no effect. No feedback is sent to the user.

See also

Programming (Page 110)

2.2.4.8 Supplementary conditions

Axis mode

If a container axis in axis mode or as a positioning spindle (POSA, SPOSA) traverses, the axis container rotates only after reaching the programmed end position.

Spindle

- A container axis active as a spindle continues to rotate during an axis container rotation.
- The control type of a spindle (speed / position control) refers to the associated machine axis. The set control type "moves" with the machine axis when an axis container rotates.
- For commands that refer to the master spindle of the channel, a machine axis with the corresponding spindle number must exist in the channel at the time of execution of the command:

  MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[<axis>] == number of the master spindle

  Note

  It is the sole responsibility of the user / machine manufacturer to ensure for spindles as container axes that an appropriate machine axis must still exist in the channel after an axis container rotation for the master spindle.

Work offsets

Note

It is the sole responsibility of the user / machine manufacturer to ensure that after an axis container rotation the effective work offsets in the channel are adapted to the changed machine axis assignments.

Continuous-path mode

If continuous-path mode is active in the channel and an axis container rotation is performed, a subsequent programming of a container axis interrupts the continuous-path mode. The interruption also occurs even when the container axis not a path axis.

PLC axis

If a container axis whose axis container has been enabled for rotation becomes a PLC axis, the status change occurs only after completion of the axis container rotation.

Command axis

If a container axis whose axis container has been enabled for rotation traverses as a command axis, the traversing movement is performed only after completion of the axis container rotation.

Oscillating axis

If a container axis whose axis container has been enabled for rotation becomes an oscillation axis, the status change occurs only after completion of the axis container rotation.

External work offset

The "external work offset" is based on the machine coordinate system (MCS). Therefore, for an active "external work offset", one of the container axes rejects the axis container rotation with alarm 4022.

Axial frames

The axial frame of a channel axis, which is also a container axis, is no longer valid after an axis container rotation. Since the axis container rotation assigns a new machine axis to the channel axis, but the axial frame is referred to a machine axis, the rotation also changes the axial frame. If the two frames do not coincide, a synchronization process (internal REORG) is performed.

Note

The assignment between a channel axis and a machine axis is altered by the axis container rotation. The current frames remain unchanged after a rotation. The user is responsible for ensuring that the correct frames are selected after a rotation by programming basic frame masks, for example.

Transformation

If a container axis is involved as a spindle in a transformation, the transformation must be deselected before the enable of the axis container rotation.

Axis couplings

If an axis coupling is active for a container axis, the coupling with `COUPOF` must be deselected before the enable of the axis container rotation. After completion of the rotation, the coupling can be immediately selected again with `COUPON`. A new define of the coupling is not required.

Gantry axis

A gantry axis cannot be a container axis.

Travel to fixed limit

If a container axis is at the limit stop, no axis container rotation can be performed.

Drive alarms

If a drive alarm is pending for a container axis, the axis container rotation is not performed.

2.2.5 Lead link axes

2.2.5.1 General information

If, for an axis coupling, the the machine axes of the leading and following axes are not connected to the same NCU, the coupling must be established using a link axis of the NCU of the following axis. In this case, the link axis is designated as lead-link axis.

The setpoints of the master axis are transferred synchronously in the interpolator cycle via the NCU link to the lead-link axis. Similarly, in the opposite direction, the actual values and the status data of the lead-link and the following axis are transferred to the leading axis.

The lead-link axis is parameterized as local leading axis of the following axis.

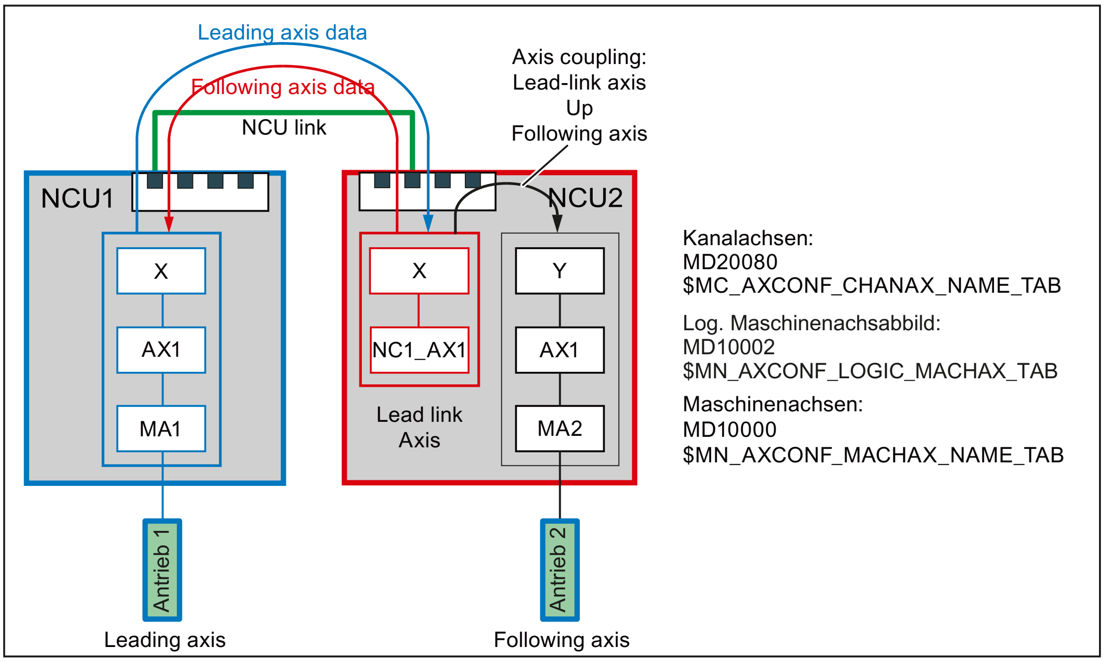


Figure 2-20 Lead-link axis

Coupled axes

Lead link axes can be used in conjunction with following axis couplings:

- Master value coupling
- Coupled motion
- Tangential tracking
- Electronic gear (ELG)
- Synchronous spindle

Requirement

The NCU must communicate via the NCU link. See Section "Link communication (Page 79)"

2.2.5.2 Parameterization

Link communication

NC-specific machine data

Number	Identifier $MN_	Meaning
MD12510	NCU_LINKNO	Unique numerical identification of the NCU within the link group. The identifiers must be assigned without any gaps in ascending ordering starting from 1. Value range: 1, 2, ... Maximum NCU number **Note:** The NCU to which the value **1** is assigned as NCU identification is the **master** NCU of the link group. The parameterization of link axes and axis containers may only be made with the machine and setting data of the master NCU.
MD18780	MM_NCU_LINK_MASK.Bit 0	Activation of the link communication
MD18781	NCU_LINK_CONNECTIONS	Number of internal link connections **Note:** It is recommended to retain the default value 0 (determination of the number by the NC).
MD18782	MM_LINK_NUM_OF_MODULES	Number of NCUs to be connected with one another via NCU link.

Setpoint synchronization

NC-specific machine data

Number	Identifier $MN_	Meaning
MD18720	MM_SERVO_FIFO_SIZE	Size of the IPO/SERVO data buffer The transfer of the setpoints of the leading axis via the NCU link to the NCU of the lead-link axis produces a deadtime of two interpolator cycles. The dead time is compensated by parameterizing the sizes of the IPO/SERVO data buffer on the NCU of the leading axis and the NCU of the lead-link axis: • NCU of the **leading axis**: 4 • NCU of the **lead-link axis**: 2

Note

With the simultaneous use of **lead link axes** and **cross-NCU axis containers**, the axis container means machine data MD18720 $MN_MM_SERVO_FIFO_SIZE = 3 must be set. This means a synchronous output of the setpoints for leading and following axis is not possible. The offset then has the size of one interpolator cycle.

Leading, lead-link and following axis

NC-specific machine data

Number	Identifier $MN_	Meaning
MD10000	AXCONF_MACHAX_NAME_TAB	Machine axis name
MD10002	AXCONF_LOGIC_MACHAX_TAB	Logical machine axis image

Channel-specific machine data

Number	Identifier $MC_	Meaning
MD20070	AXCONF_MACHAX_USED	Used machine axis

Axis-specific machine data

Number	Identifier $MA_	Meaning
MD30554	AXCONF_ASSIGN_MASTER_NCU	Master NCU If a machine axis can be traversed for several NCUs, one NCU must be defined as master NCU. The setpoint generation is made after the startup of the control for this NCU.

2.2.5.3 System variables to enter a leading value

Leading values can be specified on the NCU of the leading axis using the following system variable:

- Position leading value: $AA_LEAD_SP[<leading axis>]
- Velocity leading value: $AA_LEAD_SV[<leading axis>]

When making a change, the values are also transferred to the NCU of the following axis per NCU link.

Note

These system variables have a lower transfer priority than those of the link variables.

2.2.5.4 Supplementary conditions

The following supplementary conditions must be observed:

- The leading axis cannot be a link axis
- The leading axis cannot be a container axis
- The leading axis cannot be a gantry axis
- The leading axis may only be replaced within its own NCU (see Section "Axis/spindle replacement (Page 320)")
- Couplings with lead link axes must not be cascaded
- A lead link axis must not be traversed independently of the leading axis

---

Note

"Lead link axes and "link axes" functions

Because the "lead link axes and "link axes" functions require different settings in machine data: MD18720 $MN_MM_SERVO_FIFO_SIZE, they cannot be used simultaneously within a link group.

2.2.5.5 Example

A detailed example for parameterizing and programming an axis coupling with lead-link axis is provided in the Section: "Examples" > "Lead link axis (Page 135)".

2.2.6 System of units within a link group

For a cross-NCU interpolation, the same system of units must be active on all NCUs of the link group.

Common system of units changeover via HMI

The following conditions must be fulfilled for all NCUs of the link group in order that a system of units changeover can be made from the HMI user interface of an NCU of the link group as well as on all other NCUs of the link group:

- MD10260 $MN_CONVERT_SCALING_SYSTEM = 1
- For all channels:
  MD20110 $MC_RESET_MODE_MASK, Bit 0 = 1
- All channels are in the reset state
- No axis is traversed in the JOG or DRF mode or via the PLC
- The function "constant grinding wheel peripheral speed (GWPS)" is not active.

If, on one NCU of the link group, one of the specified conditions is not fulfilled, then the system of units changeover is not made on any of the NCUs of the link group.

Different systems of units

Different systems of units are possible in spite of an active link group, as long as no cross NCU interpolation takes place. The system of units settings are made for a specific NCU in the part program or synchronous action using G commands (`G70`, `G71`, `G700`, `G710`).

References

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

2.3 Examples

2.3.1 Link axis

Parameter example for two NCUs each with a link axis

NCU1

Machine data	Note
General link data:	
$MN_NCU_LINKNO = 1	Master NCU
$MN_MM_NCU_LINK_MASK = 1	Set NCU-link active
$MN_MM_SERVO_FIFO_SIZE = 3	Size of the data buffer between interpolation and position control
$MN_MM_LINK_NUM_OF_MODULES = 2	Number of link modules
Logical machine axis image (LAI):	
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"	Local machine axis
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"	Local machine axis
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX3"	Link axis
Machine axis name, unique system-wide as NCU identifier:	
$MN_AXCONF_MACHAX_NAME_TAB[0] = "NC1_A1"	
$MN_AXCONF_MACHAX_NAME_TAB[1] = "NC1_A2"	
$MN_AXCONF_MACHAX_NAME_TAB[2] = "NC1_A3"	
Assignment of channel axis to machine axis:	
$MC_AXCONF_MACHAX_USED[0] = 1	1. Channel axis to the machine axis of LAI[0]
$MC_AXCONF_MACHAX_USED[1] = 2	2. Channel axis to the machine axis of LAI[1]
$MC_AXCONF_MACHAX_USED[2]=3	3. Channel axis to the machine axis of LAI[2]

NCU2

Machine data	Note
General link data:	
$MN_NCU_LINKNO = 2	Slave NCU
$MN_MM_NCU_LINK_MASK = 1	Set NCU-link active
$MN_MM_SERVO_FIFO_SIZE = 3	Size of the data buffer between interpolation and position control
$MN_MM_LINK_NUM_OF_MODULES = 2	Number of link modules
Logical machine axis image (LAI):	
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"	Local machine axis
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"	Local machine axis
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC1_AX3"	Link axis
Machine axis name, unique system-wide as NCU identifier:	
$MN_AXCONF_MACHAX_NAME_TAB[0] = "NC2_A1"	
$MN_AXCONF_MACHAX_NAME_TAB[1] = "NC2_A2"	
$MN_AXCONF_MACHAX_NAME_TAB[2] = "NC2_A3"	
Assignment of channel axis to machine axis:	
$MC_AXCONF_MACHAX_USED[0] = 1	1. Channel axis to the machine axis of LAI[0]
$MC_AXCONF_MACHAX_USED[1] = 2	2. Channel axis to the machine axis of LAI[1]
$MC_AXCONF_MACHAX_USED[2]=3	3. Channel axis to the machine axis of LAI[2]

2.3.2 Axis container coordination

The characteristic as a function of time is displayed from top to bottom in the following tables. The data are valid on condition that only two channels have axes in the container.

2.3.2.1 Axis container rotation without a part program wait

Channel 1	Channel 2	Comment
AXCTWE(C1)	Part program ...	Channel 1 enables the axis container for rotation.
Part program **without** movement of a container axis	Part program ...	
	AXCTSWE(C1)	Channel 2 enables the axis container for rotation, container rotates because both channels have enabled rotation
Part program **with** movement of a container axis	Part program **with** movement of a container axis	**Without wait**

2.3.2.2 Axis container rotation with an implicit part program wait

Channel 1	Channel 2	Comment
AXCTWE(C1)	Part program ...	Channel 1 enables the axis container for rotation.
Part program **with** movement of a container axis	Part program ...	Channel 1 **waits implicitly** for axis container rotation
	AXCTSWE(C1)	Channel 2 enables the axis container for rotation, rotation occurs. Channel 1 is continued.

2.3.2.3 Axis container rotation by one channel only (e.g. during power up)

Channel 1	Channel 2	Comment
AXCTWE**D**(C1)	In the RESET state	Instantaneous rotation

2.3.3 Evaluating axis container system variables

2.3.3.1 Conditional branch

Channel 1	Comment
N100 AXCTWE(CT1)	Channel 1: Enable of the rotation of axis container CT1
MARKER1:	
N200 ...	Part program **without** movement of a container axis
IF $AC_AXCTSWA[CT1] == 1 GOTOB MARKE1	IF rotation of axis container CT1 still active THEN continue with MARKER1 ELSE (rotation of axis container CT1 completed)
N300 ...	Part program **with** movement of a container axis

2.3.3.2 Static synchronized action with $AN_AXCTSWA

Channel 1	Comment
IDS =1 EVERY $AN_AXCTSWA[CT1] == 1 DO M99	Static synchronized action: Always output auxiliary function M99 at the beginning of an axis container rotation.
	References: Synchronized Actions Function Manual

2.3.3.3 Wait for certain completion of axis container rotation

If you want to wait until the axis container rotation is reliably completed, you can use one of the examples below selected to suit the relevant situation.

Example 1

rl = $AN_AXCTAS[ctl]; Read current axis container position

AXCTSWE(ctl) ; Permit axis container rotation

WHILE (rl == $AN_AXCTAS[ctl]); Wait until axis container position

ENDWHILE ; has changed

Example 2 for 1st channel

CLEARM(9); Delete synchronization marker 9

AXCTSWE(ctl) ; Permit axis container rotation

; wait with synchronized action until

; axis container rotation is completed

WHEN $AN_AXCTSWA[ctl] == TRUE DO SETM(9) ; Set marker 9 and

WAITMC(9, 1) ; Wait for synchronization marker 9

; in first channel

Example 3.1 Use internal wait

M3 S100 ; Reprogram axis container **spindle**

; An internal wait takes place for the end of

; axis container rotation

Example 3.2 Use internal wait

x=IC(0) ; Reprogram axis container **axis** x

; An internal wait takes place for the end of

; axis container rotation

Example 3.3 Use internal wait

AXCTSWE(CTL) ; If an axis container is reenabled for rotation,

; an internal wait takes place for the end of the earlier

; axis container rotation.

N2150 WHILE (rl == $AN_AXCTAS[ctl])

Note

Programming in the NC program:

WHILE ($AN_AXCTSWA[n] == 0)

ENDWHILE

cannot be used as a reliable method of determining whether an earlier axis container rotation has finished. Although in software version 7.x and later, $AN_AXCTSWA performs an implicit preprocessing stop, this type of programming cannot be used, as the block can be interrupted by a reorganization. The system variable then returns "0" as the axis container rotation is then ended.

2.3.4 Configuration of a multi-spindle turning machine

Introduction

The following example describes the use of:

- Several NCUs in the NCU link group
- Flexible configuration with axis containers

Machine description

- Distributed on the circumference of a drum A (front-plane machining) the machine has:
  - 4 main spindles, HS1 to HS4

    Each main spindle has the possibility of material feed (bars, hydraulic bar feed, axes: STN1 - STN4).
  - 4 cross slides
  - Each slide has two axes.
  - Optionally a powered tool S1-S4 can operate on each slide.

- Distributed on the circumference of a drum B (rear-plane machining) the machine has:
  - 4 counterspindles GS1 to GS4
  - 4 cross slides
  - Each slide has two axes.
  - Optionally a powered tool S5-S8 can operate on each slide.
  - The position of each counterspindle can be offset through a linear axis for example for transferring parts from the main spindle for rear-plane machining in drum B. (Transfer axes. Axes: ZG1 - ZG4).
- Couplings:
  - If drum A rotates, all main spindles of this drum are subordinate to another group of slides.
  - If drum B rotates, all main counterspindles and all transfer axes of this drum are subordinate to another group of slides.
  - The rotations of drums A and B are autonomous.
  - The rotations of drums A and B are limited to 270 degrees.

    (range and twisting of supply cables)

Term: Position

Main spindle HS_i and counterspindle GS_i together with their slides characterize a position.

NCU assignment

The axes and spindles of a position (for this example) are each assigned to an NCU. One of the NCUs, the master NCU, controls the axes for the rotations of drums A and B additionally. There are 4 NCUs with a maximum of the following axes:

Number of axes

Per NCU_i the following axes/spindles must be configured:

Slide 1: X_i1, Z_i1

2: X_i2, Z_i2

Spindles: HS_i, GS_i, powered tools: S1, S2

Transfer axis: ZG_i

Bar feed: STN_i.

For the master NCU, in addition to the above-mentioned axes there are the two axes for rotating drums A and B. The list shows that it would not be possible to configure the axis number for a total of 4 positions via an NCU. (Limit 31 axes, required are 4 + 10 + 2 axes).

Axis container

With rotation of drums A/B, HS_i, GS_i, ZG_i and STN_i must be assigned to another NCU and must therefore be configured as link axes in axis containers.

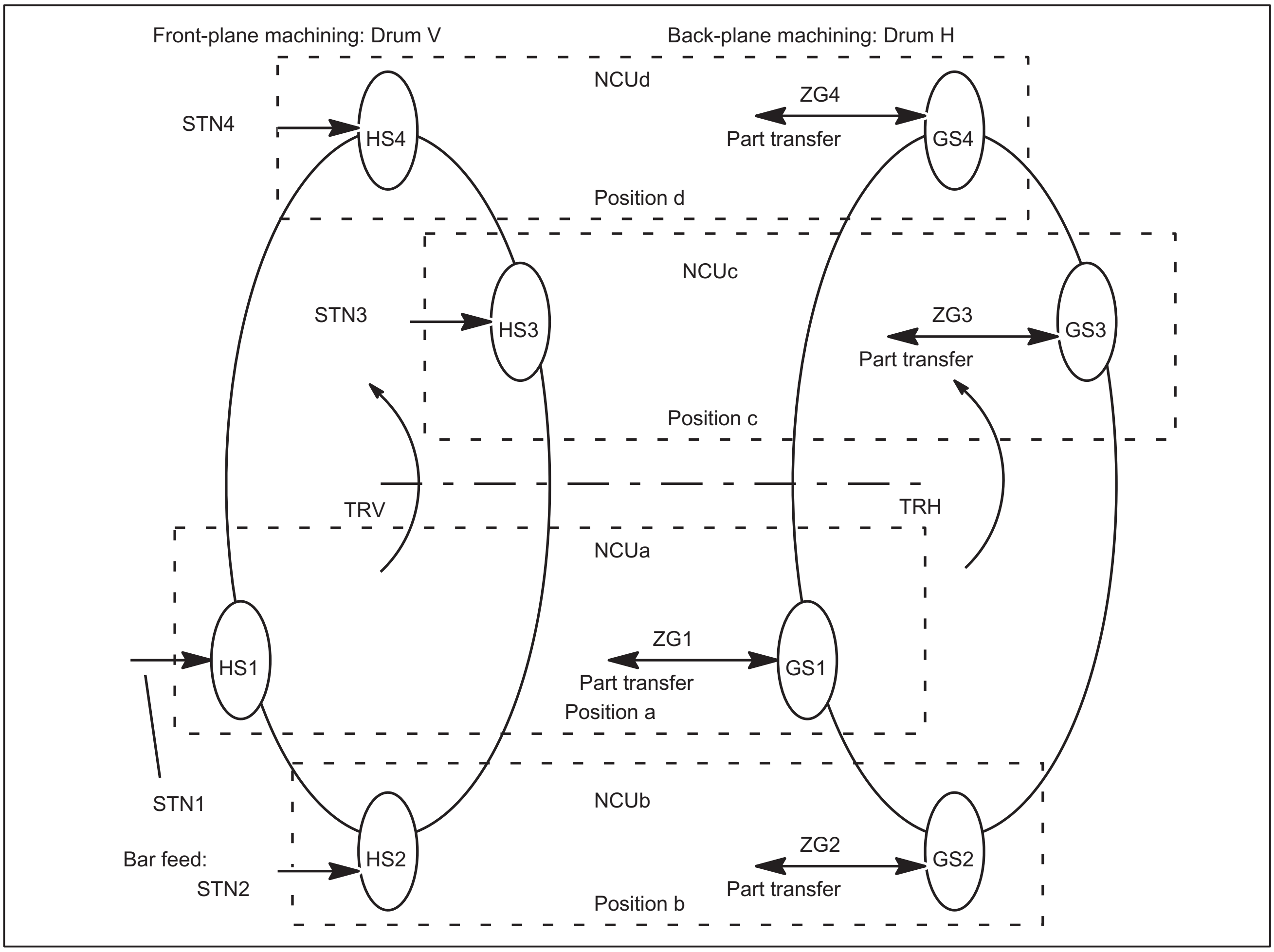


Figure 2-21 Schematic diagram of main spindles HSi, countersp. GSi, bar feed axis STNi and transfer axes ZGi

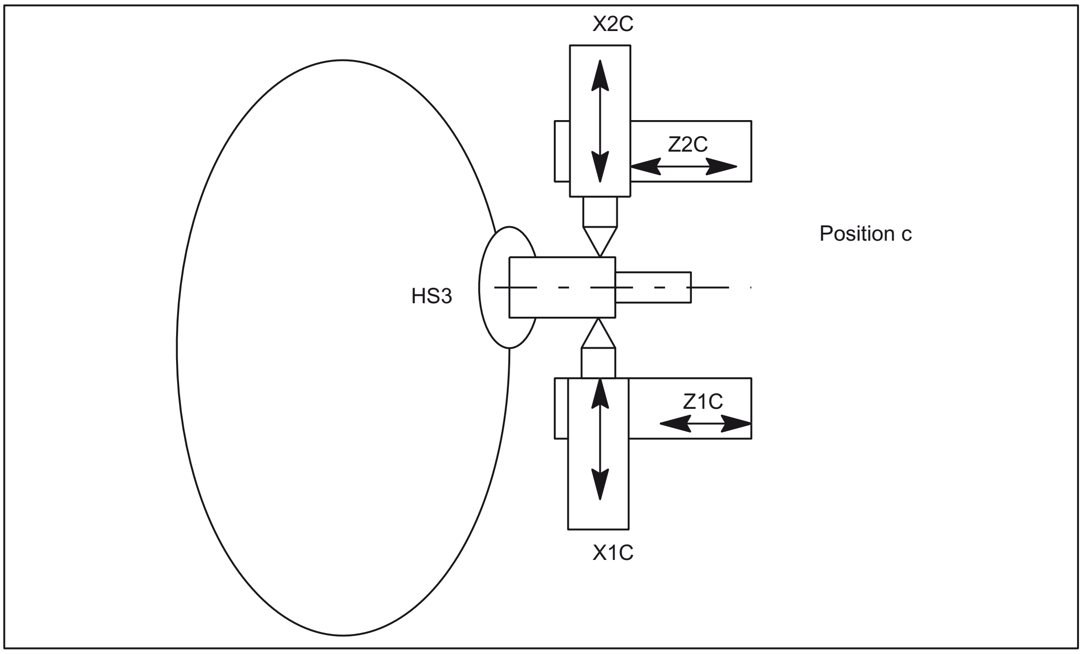


Figure 2-22 Two slides per position can also operate together on one spindle.

Note

The axes are given the following names in order to clarify the assignments of axes to slides and positions:

Xij with i slide (1, 2), j position (A-D)

Zij with i slide (1, 2), j position (A-D)

Whereas the positions and their slides remain in one place, main spindles, counterspindles, bar feed axes STN and transfer axes ZG move to new positions as the result of rotations of drums V or H.

For example, the axes to be managed per NC when the slide is taken into account are as follows for the configurations shown in the above diagrams:

Axes of master NCU

Table 2- 1 Axes of master NCU: NCUa

Common axes	Local axes	Comment
	TRV (drum V)	Master NCU only
	TRH (drum H)	Master NCU only
	X1A	Slide 1
	Z1A	Slide 1
	X2A	Slide 2
	Z2A	Slide 2
	S1	Slide 1
	S2	Slide 2
HS1		Axis container necessary
GS1		Axis container necessary
ZG1		Axis container necessary
STN1		Axis container necessary
4	8	

Axes of NCUb to NCUd

The NCUs that are not master NCUs have the same axes with the exception of the axes for the drive for drums TRV and TRH. The letter designating the position must be replaced accordingly for the NCU and axis name (a, A → b, B to d, D).

Configuration rules

The following rules were applied for the configuration described below:

- Main spindle, counterspindles and axes that are assigned to different NCUs through drum rotation while they are operating as illustrated in the above diagram "Main spindle ..." must be configured in an axis container.

  (HS_i, GS_i, ZG_i, STN_i).
- All main spindles for drum A are in the same container (No. 1).
- All bar feed axes for drum A are in the same container (No. 2).
- All counterspindles for drum B are in the same container (No. 3).
- All transfer axes for drum B are in the same container (No. 4).
- Main spindles HS_i and their counterspindle GS_i as well as the transfer axes for counterspindle ZG_i and the bar feed axes STN_i of the main spindle are assigned as follows for uniform load distribution purposes:

  NCUa HS1-STN1,

  NCUb HS2-STN2, ... etc.
- Slide axes Xij, Zij are solely local axes with a fixed NCU assignment.
- Slides are assigned to a dedicated channel of an NCU.

  Slides can therefore be moved autonomously.

Configuration options

- Main or counterspindles are flexibly assigned to the slide.
- The speed of the main spindle and the counterspindle can be defined independently in each position.

  Exceptions:

  During the parts change from front-plane machining in drum V to rear-plane machining in drum H, the speeds of the main spindle and the counterspindle must be synchronized (synchronous spindle coupling).

  In cases where slide 2 also participates in front-plane machining so as to "support" slide 1, the main spindle speed also applies to slide 2. Similarly if slide 1 participates in rear-plane machining, the counterspindle speed also applies to slide 1.

Minor changes in speed

Due to the unavoidable time delays incurred in the processing of actual values, abrupt changes in speed should be avoided during cross-NCU machining operations. Compare axis data and signals.

Configuration for NCU1

Uniform use of channel axis names in the part programs:

S4: Main spindle

S3: Counterspindle

X1: Infeed axis

Z1: Longitudinal axis

S1: Powered tool

Z3: Transfer axis

TRV: Drum V for main spindle

TRH: Drum H for counterspindle

STN: Hydraulic bar feed

Axes highlighted in **bold** characterize the current channel as home channel for the axis in conjunction with axis replacement.

Table 2- 2 NCUa, position: a, channel: 1, slide: 1

Channel axis name	..._MACHAX _USED	$MN_ AXCONF_LOGIC_MACHAX_TAB,	Container, slot entry (string)	Machine axis name
S4	1	AX1: CT1_SL1	1 1 NC1_AX1	HS1
S3	2	AX2: CT3_SL1	3 1 NC1_AX2	GS1
X1	3	AX3:		X1A
Z1	4	AX4:		Z1A
Z3	5	AX5: CT4_SL1	4 1 NC1_AX5	ZG1
S1	*6*	*AX6:*		*WZ1A*
STN	7	AX7: CT2_SL1	2 1 NC1_AX7	STN1
TRV	11	AX11:		TRV
TRH	12	AX12:		TRH
x2 *				
z2 *				

Table 2- 3 NCUa, position: a, channel: 2, slide: 2

Channel axis name	..._MACHAX _USED	$MN_ AXCONF_LOGIC_MACHAX_TAB,	Container, slot entry (string)	Machine axis name
S4	1	AX1: CT1_SL1	1 1 NC1_AX1	HS1
S3	2	AX2: CT3_SL1	3 1 NC1_AX2	GS1
Z3	5	AX5: CT4_SL1	4 1 NC1_AX5	ZG1
STN	7	AX7: CT2_SL1	2 1 NC1_AX7	STN1
X2	8	AX8:		X2A
Z2	9	AX9:		Z2A
S1	*10*	*AX10:*		*WZ2A*
x1 *				
z1 *				

Note

* due to program coordination via axis positions and 4-axis machining in one position

Entries in the axis container locations should have the following format: "NC1_AX.." with the meaning NC1 = NCU 1. In the above tables, NCUa is imaged on NC1_..., NCUb on NC2_... etc.

Further NCUs

The above listed configuration data must be specified accordingly for NCUb to NCUd. Please note the following:

- Axes TRA and TRB only exist for NCUa, channel 1.
- The container numbers are maintained for the other NCUs as they were specified for the individual axes
- The slot numbers are as follows:

  NCUb → 2

  NCUc → 3

  NCUd → 4.
- The machine axis names are as follows:

  NCUb → HS2, GS2, ZG2, STN2

  NCUc → HS3, GS3, ZG3, STN3

  NCUd → HS4, GS4, ZG4, STN4.

Axis container

The information relating to containers given in Table 7-17 and the container entries of the similarly configured NCUs, NCUb to NCUd, are specified in the following tables, sorted according to containers and slots, as they have to be set in machine data:

MD12701 $MN_AXCT_AXCONF_ASSIGN_TAB1[slot]

. . .

MD12716 $MN_AXCT_AXCONF_ASSIGN_TAB16[slot]

whereby slots: 1-4 must be set for the 4 positions of a multi-spindle turning machine:

Note

For the machine data entry

$MN_AXC_AXCONF_ASSIGN_TAB$_i$[slot]

the values (without decimal point and machine axis name) that are entered under initial position in the above tables must be set.

Table 2- 4 Axis container and their position-dependent contents for drum A

Container	Slot	Initial position (TRA 0°)	Switch 1 (TRA 90°)	Switch 2 (TRA 180°)	Switch 3 (TRA 270°)	Switch 4 = (TRA 0°)
1	1	NC1_AX1, HS1	NC2_AX1, HS2	NC3_AX1, HS3	NC4_AX1, HS4	NC1_AX1, HS1
	2	NC2_AX1, HS2	NC3_AX1, HS3	N4C_AX1, HS4	NC1_AX1, HS1	NC2_AX1, HS2
	3	NC3_AX1, HS3	NC4_AX1, HS4	NC1_AX1, HS1	NC2_AX1, HS2	NC3_AX1, HS3
	4	NC4_AX1, HS4	NC1_AX1, HS1	NC2_AX1, HS2	NC3_AX1, HS3	NC4_AX1, HS4
2	1	NC1_AX7, STN1	NC2_AX7, STN2	NC3_AX7, STN3	NC4_AX7 STN4	NC1_AX7, STN1
	2	NC2_AX7, STN2	NC3_AX7, STN3	NC4_AX7, STN4	NC1_AX7, STN1	NC2_AX7, STN2
	3	NC3_AX7, STN3	NC4_AX7, STN4	NC1_AX7, STN1	NC2_AX7, STN2	NC3_AX7, STN3
	4	NC4_AX7, STN4	NC1_AX7, STN1	NC2_AX7, STN2	NC3_AX7, STN3	NC4_AX7, STN4
Drum movement		0°	+ 90°	+ 90°	+ 90°	- 270°

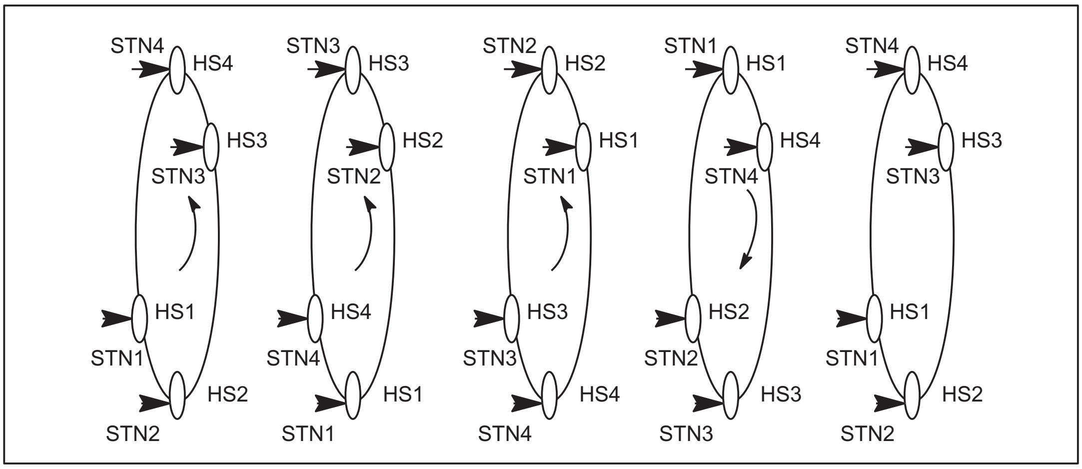


Figure 2-23 Positions of drum A

Table 2- 5 Axis container and their position-dependent contents for drum B

Container	Slot	Initial position (TRB 0°)	Switch 1 (TRB 90°)	Switch 2 (TRB 180°)	Switch 3 (TRB 270°)	Switch 4 = (TRB 0°)
3	1	NC1_AX2, GS1	NC2_AX2, GS2	NC3_AX2, GS3	NC4_AX2, GS4	NC1_AX2, GS1
	2	NC2_AX2, GS2	NC3_AX2, GS3	NC4_AX2, GS4	NC1_AX2, GS1	NC2_AX2, GS2
	3	NC3_AX2, GS3	NC4_AX2, GS4	NC1_AX2, GS1	NC2_AX2, GS2	NC3_AX2, GS3
	4	NC4_AX2, GS4	NC1_AX2, GS1	NC2_AX2, GS2	NC3_AX2, GS3	NC4_AX2, GS4
4	1	NC1_AX5, ZG1	NC2_AX5, ZG2	NC3_AX5, ZG3	NC4_AX5 ZG4	NC1_AX5, ZG1
	2	NC2_AX5, ZG2	NC3_AX5, ZG3	NC4_AX5, ZG4	NC1_AX5, ZG1	NC2_AX5, ZG2
	3	NC3_AX5, ZG3	NC4_AX5, ZG4	NC1_AX5, ZG1	NC2_AX5, ZG2	NC3_AX5, ZG3
	4	NC4_AX5, ZG4	NC1_AX5, ZG1	NC2_AX5, ZG2	NC3_AX5, ZG3	NC4_AX5, ZG4

2.3.5 Lead link axis

2.3.5.1 Configuration

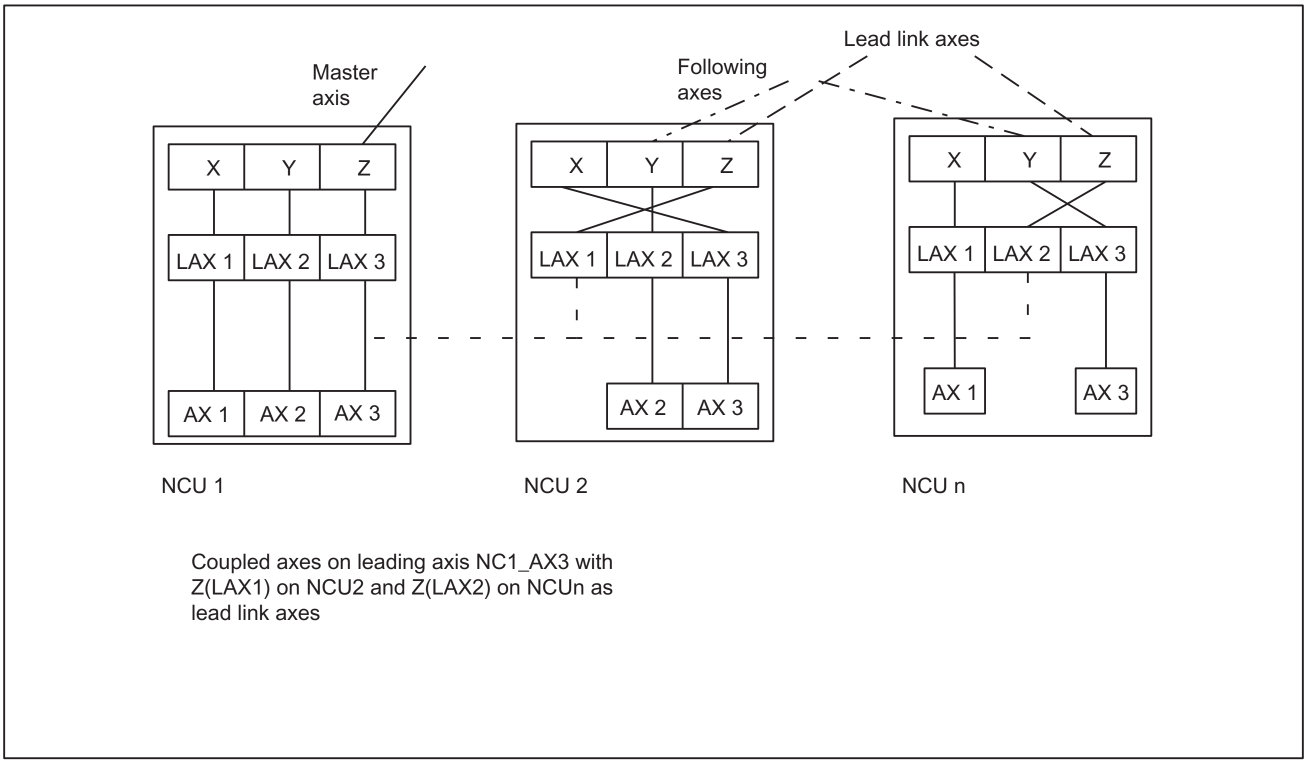


Figure 2-24 NCU2 to NCUn use a lead link axis to enable coupling to the machine axis on NCU1 (NCU1-AX3).

The following example refers to the axis coupling section between Y(LAX2, AX2) as following axis on NCU2 and Z(LAX3, NC1_AX3) as lead link axis.

Machine data

- The machine data of a leading value axis may only be loaded on the home NCU. From this NCU, the relevant machine data are distributed to the other NCUs where a lead link axis has been defined.
- Each lead link axis reduces the maximum number of axes that can be traversed on this NCU by one axis.

Machine data for NCU1 (leading axis)

Machine data	Meaning
$MN_NCU_LINKNO = 1	1. or master NCU
$MN_MM_NCU_LINK_MASK = 1	NCU link active
$MN_MM_LINK_NUM_OF_MODULES= 2	Number of link modules
$MN_MM_SERVO_FIFO_SIZE = 4	The size of the data buffer is increased to 4 between interpolation and position control
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"	
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"	
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "AX3"	
$MN_AXCONF_MACHAX_NAME_TAB[0] = "XM1"	
$MN_AXCONF_MACHAX_NAME_TAB[2] = "YM1"	
$MA_AXCONF_ASSIGN_MASTER_NCK[AX3] = 1	
$MC_AXCONF_MACHAX_USED[0]=1 ; X	
$MC_AXCONF_MACHAX_USED[1]=2 ; Y	
$MC_AXCONF_MACHAX_USED[2]=3 ; Z	

Machine data for NCU2 (following axis)

Machine data	Meaning
$MN_NCU_LINKNO = 2	2. NCU number
$MN_MM_NCU_LINK_MASK = 1	Activate link
$MN_MM_NUM_CURVE_TABS = 5	Number of curve tables
$MN_MM_LINK_NUM_OF_MODULES= 2	Number of link modules
$MN_MM_NUM_CURVE_SEGMENTS= 50	
$MN_MM_NUM_CURVE_POLYNOMS = 100	
$MN_MM_SERVO_FIFO_SIZE = 2	Size of the data buffer between interpolation and position control (standard)
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "NC1_AX3"	Lead link on NCU1/AX3
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"	
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "AX3"	
$MC_AXCONF_MACHAX_USED[0]=3	AX3 is machine axis of the 1st channel axis
$MC_AXCONF_MACHAX_USED[1]=2	AX2 is machine axis of the 2nd channel axis
$MC_AXCONF_MACHAX_USED[2]=1	AX3 on NCU1 is machine axis of the 3rd channel axis

2.3.5.2 Programming

Program for NCU1 (leading axis)

NCU1 traverses leading axis Z

Identifier for NCU2, that the leading axis of NCU1 is assigned: Link variable $A_DLB[0] = 1

Identifier for NCU2, that the leading axis of NCU1 has been released: Link variable $A_DLB[0] = 0

Program code	Comment
N1000 R1 = 0	; Initialize loop counter
N1004 G1 Z0 F1000	; Traverse axis Z to the starting position
N1005 $A_DLB[0] = 1	; Identifier for NCU2: Axis Z is assigned
LOOP10:	;
N1005 R1=R1+1	; Increment loop counter
N1006 Z0.01 G91	; Traversing the leading value axis Z
N1008 Z0.02	; Traversing the leading value axis Z
N1010 Z0.03	; Traversing the leading value axis Z
N1012 IF R1 < 10 GOTOB LOOP10	;
N1098 $A_DLB[0] = 0	; Identifier for NCU2: Axis Z is free

Program for NCU2 (following axis)

The program establishes a connection between leading axis movements on NCU1 and following axis movements on NCU2 via a curve table. If the table has been defined, NCU2 goes into the wait position (N2006) until NCU1 has assigned axis Z as the leading axis (N1005). The coupling is activated (N2010) as soon as axis Z has been assigned as leading axis. The coupling is kept until NCU1 has released axis Z as the leading axis.

Program code	Comment
N2000 CTABDEL(1)	; Initialize table 1
N2001 G04 F.1	;
N2003 G0 Y0 Z0	; Traverse axes Y, Z into the starting position
N2002 CTABDEF(Y, Z, 1, 0)	; Table definition ON
N2003 G1 X0 Y0	; Intermediate point 1
N2004 G1 X100 Y200	; Intermediate point 2
N2005 CTABEND	; Table definition OFF
LOOP20:	;
N2006 IF ($A_DLB[0] == 0) GOTOB LOOP20	; Wait for NCU1
N2010 LEADON(Y,Z,1)	; => close coupling
LOOP25:	;
N2030 IF ($A_DLB[0] > 0) GOTOB LOOP25	; Keep the coupling until NCU1 no longer traverses the leading value axis
N2090 LEADOF(Y,Z)	; => open coupling

2.4 Data lists

2.4.1 Machine data

2.4.1.1 General machine data

Number	Identifier: $MN_	Description
10002	AXCONF_LOGIC_MACHAX_TAB	Logical NCU machine axis image
10065	POSCTRL_DESVAL_DELAY	Position setpoint delay
10134	MM_NUM_MMC_UNITS	Number of simultaneous MMC communication partners
12510	NCU_LINKNO	NCU number in an NCU group
12520	LINK_TERMINATION	NCU numbers for which bus terminating resistors are active
12530	LINK_NUM_OF_MODULES	Number of link modules
12701 ... 12716	AXCT_AXCONF_ASSIGN_TAB1 ... AXCT_AXCONF_ASSIGN_TAB16	List of axes in the axis container
12750	AXCT_NAME_TAB	List of axis container names
12760	AXCT_FUNCTION_MASK	Functions for the axis container
18700	MM_SIZEOF_LINKVAR_DATA	Size of the link variables memory
18720	MM_SERVO_FIFO_SIZE	Size of the data buffer between interpolation and position controller
18780	MM_NCU_LINK_MASK, bit 0	Link communication activation

2.4.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20000	CHAN_NAME	Channel name
20070	AXCONF_MACHAX_USED	Machine axis number valid in channel
28160	MM_NUM_LINKVAR_ELEMENTS	Number of write elements for the NCU link variables

2.4.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30550	AXCONF_ASSIGN_MASTER_CHAN	Default assignment between an axis and a channel
30554	AXCONF_ASSIGN_MASTER_NCU	Initial setting defining which NCU generates setpoints for the axis
30560	IS_LOCAL_LINK_AXIS	Axis is a local link axis
32990	POCTRL_DESVAL_DELAY_INFO	Current position setpoint delay

2.4.2 Setting data

2.4.2.1 General setting data

Number	Identifier: $SA	Description
41700	AXCT_SWWIDTH[container number]	Axis container rotation setting

2.4.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43300	ASSIGN_FEED_PER_REV_SOURCE	Rotational feedrate for positioning axes/spindles

2.4.3 Signals

2.4.3.1 Signals from NC

Signal name	SINUMERIK 840D sl
MCP1 ready	DB10.DBX104.0
MCP2 ready	DB10.DBX104.1
HHU ready	DB10.DBX104.2
NCU link active	DB10.DBX107.6
HMI2-CPU ready (HMI connected to OPI or MPI)	DB10.DBX108.1
HMI1-CPU at MPI ready	DB10.DBX108.2
HMI1-CPU at OPI ready (standard connection)	DB10.DBX108.3

2.4.3.2 Signals from HMI/PLC

Signal name	SINUMERIK 840D sl
ONL_REQUEST Online request from HMI	DB19.DBB100
ONL_CONFIRM Acknowledgement from PLC for online request	DB19.DBB102
PAR_CLIENT_IDENT HMI writes its client identification (bus type, HMI bus address)	DB19.DBB104
PAR_MMC_TYP HMI type according to NETNAMES.INI: Main/secondary operator panel/alarm server	DB19.DBB106
PAR_MSTT_ADR HMI writes address to the MCP to be activated	DB19.DBB107
PAR_STATUS PLC writes the online enable for the HMI (connection state)	DB19.DBB108
PAR_Z_INFO PLC writes additional information to the connection state	DB19.DBB109
M_TO_N_ALIVE Sign of life from the PLC to HMI using M to N block	DB19.DBB110

2.4.3.3 General online interface

Signal name	SINUMERIK 840D sl
MMC1_CLIENT_IDENT PLC writes PAR_CLIENT_IDENT to MMCx_CLIENT_IDENT, if HMI goes online.	DB19.DBB120
MMC1_TYP PLC writes PAR_MMC_TYP to MMCx_TYP, if HMI goes online.	DB19.DBB122
MMC1_MSTT_ADR PLC writes PAR_MSTT_ADR to MMCx_MSTT_ADR, if HMI goes online	DB19.DBB123
MMC1_STATUS Connection state, HMI and PLC write alternating, their requests/acknowledgements.	DB19.DBB124
MMC1_Z_INFO Additional information, connection state (pos./neg. acknowledgement, error messages, ...)	DB19.DBB125
MMC1_SHIFT_LOCK HMI switchover lock	DB19.DBX126.0

Signal name	SINUMERIK 840D sl
MMC1_MSTT_SHIFT_LOCK MCP switchover lock	DB19.DBX126.1
MMC1_ACTIVE_REQ HMI requests active operator mode	DB19.DBX126.2
MMC1_ACTIVE_PERM Enable from PLC to change the operator mode	DB19.DBX126.3
MMC1_ACTIVE_CHANGED HMI has changed the operator mode	DB19.DBX126.4
MMC1_CHANGE_DENIED HMI active/passive switchover denied	DB19.DBX126.5
MMC2_CLIENT_IDENT PLC writes PAR_CLIENT_IDENT to MMCx_CLIENT_IDENT, if HMI goes online.	DB19.DBB130
MMC2_TYP PLC writes PAR_MMC_TYP to MMCx_TYP, if HMI goes online.	DB19.DBB132
MMC2_MSTT_ADR PLC writes PAR_MSTT_ADR to MMCx_MSTT_ADR, if HMI goes online.	DB19.DBB133
MMC2_STATUS Connection state, HMI and PLC write alternating, their requests/acknowledgements.	DB19.DBB134
MMC2_Z_INFO Additional information, connection state (pos./neg. acknowledgement, error messages, ...)	DB19.DBB135
MMC2_SHIFT_LOCK HMI switchover lock	DB19.DBX136.0
MMC2_MSTT_SHIFT_LOCK MCP switchover lock	DB19.DBX136.1
MMC2_ACTIVE_REQ HMI requests active operator mode	DB19.DBX136.2
MMC2_ACTIVE_PERM Enable from PLC to change the operator mode	DB19.DBX136.3
MMC2_ACTIVE_CHANGED HMI has changed the operator mode	DB19.DBX136.4
MMC2_CHANGE_DENIED HMI active/passive switchover denied	DB19.DBX136.5

2.4.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
NCU link axis active	DB31, ...DBX60.1	-
Axial alarm	DB31, ...DBX61.1	DB390x, DBX1.1
Axis ready	DB31, ...DBX61.2	DB390x, DBX1.2
Axis container rotation active	DB31, ...DBX62.7	-

2.4.4 System variables

System variable	Description
$AN_AXCTSWE[axis]	Supplies the slots of the axis container of the specified axis which are enabled for the next axis container rotation
$AN_LAI_AX_IS_AXCTAX	Contains the container axes of the logical machine axis image as bit field
$AN_LAI_AX_IS_LINKAX	Contains the link axes of the logical machine axis image as bit field
$AN_LAI_AX_IS_LEADLINKAX	Contains the lead-link axes of the logical machine axis image as bit field
$AN_LAI_AX_TO_MACHAX[axis]	For the specified axis of the logical machine axis image, supplies the NCU-ID and the axis number of the associated machine axis
$AN_LAI_AX_TO_IPO_NC_CHANAX[axis]	For the specified axis of the logical machine axis image, supplies the channel and channel axis number and/or the NCU and global axis number
$AN_IPO_CHANAX[global axis number]	For the specified global axis number, supplies the channel and channel axis number
$AA_MACHAX[axis]	For the specified axis, supplies the NCU-ID and the machine axis number
$AA_IPO_NC_CHANAX[axis]	For the specified axis, supplies the channel and channel axis number or NCU-ID and the global axis number
$VA_IPO_NC_CHANAX[axis]	For the specified machine axis, supplies the channel and channel axis number or NCU-ID and global axis number

A more detailed description of system variables can be found in
References:
System Variables List Manual

H1: Manual and handwheel travel 3

3.1 Introduction

3.1.1 Overview

Even on modern, numerically controlled machine tools, a facility must be provided that allows the user to traverse the axes manually.

- Setting up the machine

  This is especially necessary when a new machining program is being set up and the machine axes have to be moved with the traversing keys on the machine control panel or with the electronic handwheel. Where coordinate offset or rotation is selected, manual travel can even be performed in the transformed workpiece coordinate system.

- Retraction of the tool after a program abort

  After a program abort due to a power failure or RESET, the operator must manually retract the tool from its current machining position. This is usually done by operating the traversing keys in JOG mode. The transformations and coordinate systems used for machining must remain active while this is done.

Functions

The following functions are available to manually traverse axes:

- Continuous traversing in jog or continuous mode
- Incremental traversing in jog or continuous mode
- Traversing of axes via electronic handwheels
- Handwheel override in AUTOMATIC mode with path specification and/or velocity override
- Correction of the tool wear for machining in AUTOMATIC mode by means of additional incremental work offset using the handwheel (DRF)
- Approach of fixed points specified via the machine data
- Retraction movement in the tool direction after a program abort due to a power failure or RESET

3.1.2 General characteristics when traversing in the JOG mode

The following is a description of the characteristics which generally apply to manual travel in JOG mode (irrespective of the type selected).

JOG mode

Manual traversing of axes via the traversing keys of the machine control panel by the operator, referred to as manual traversing in the following, is performed in JOG mode.

If JOG mode is active for the current mode group, this is reported to the PLC via the corresponding interface signal:

DB11 DBX6.2 (BAG1: operating mode JOG)

DB11 DBX26.2 (BAG2: operating mode JOG)

...

DB11 DBX186.2 (BAG10: operating mode JOG)

References:
Function Manual, Basic Functions; Mode Group, Channel, Program Operation, Reset Response (K1)

Machine functions

There are several JOG variants, the so-called machine functions, within JOG mode:

- Continuous (JOG CONT)
- Incremental (JOG INC)
- Jogging with the handwheel

Handwheel travel

Handwheel travel is also active with the following functions:

- JOG REPOS machine function for traversing the geometry and machine axes
- AUTOMATIC mode for moving out a DRF offset
- With path override
- When moving the reversal point of an oscillation

The active machine function is selected via the PLC interface. A separate PLC interface exists for both the machine axes (axis-specific) and the geometry axes (channel-specific).

Traversing several axes

Several axes can be traversed simultaneously in JOG mode. However, there is no interpolatory relationship.

Velocity

The velocity for a JOG traversing movement is determined by the following value settings depending on the feedrate mode:

- Linear feedrate (G94) is active (SD41100 $SN_JOG_REV_IS_ACTIVE = 0):
  - With the general setting data:

    SD41110 $SN_JOG_SET_VELO (axis velocity for JOG)

    Or, for rotary axes with general setting data:

    SD41130 $SN_JOG_ROT_AX_SET_VELO
    (JOG speed for rotary axes)
  - Or (only if SD41110 = 0) with the axial machine data:

    MD32020 $MA_JOG_VELO (conventional axis velocity)
- Revolutional feedrate (G95) is active (SD41100 $SN_JOG_REV_IS_ACTIVE = 1):
  - With the general setting data:

    SD41120 $SN_JOG_REV_SET_VELO (revolutional feedrate of axes in JOG)
  - Or (only if SD41120 = 0) with axial machine data:

    MD32050 $MD_JOG_REV_VELO (revolutional feedrate for JOG)

The default setting for feedrate velocity is mm/min or rpm for revolutional feedrate or rotary axes.

Note

Because of the limited feedrate, the axis is not able to follow the handwheel rotation synchronously, especially in the case of a large pulse weighting, and therefore overtravels.

Rapid traverse override

If the traversing key is also actuated together with the rapid traverse override key, the movement will be made with the configured rapid traverse velocity:

- MD32010 $MA_JOG_VELO_RAPID (rapid traverse in jog mode)
- MD32040 $MA_JOG_REV_VELO_RAPID (rapid traverse, revolutional feedrate)

Feedrate override

The traversing velocity for JOG can also be influenced using the axial feedrate override switch provided that the following NC/PLC interface signal is set:

DB31, ... DBX1.7 (axial feedrate override active)

Percentages are assigned to the individual feedrate override switch positions via machine data. For a switch position of 0% the axis is not traversed, provided that 0 is entered in the associated machine data.

The interface signal DB31, ... DBX1.7 (axial feedrate override active) has no meaning for switch position 0%.

Instead of being set by the feedrate override switch position (gray code), the percentage value (0% to 200%) can optionally be set directly by the PLC. Again, the selection is made via machine data.

References:
Function Manual, Basic Functions; Feedrates (V1)

Acceleration

With manual travel and handwheel travel, acceleration also takes place according to a programmed characteristic. The acceleration characteristic effective for JOG for the individual axis is defined, using the following axial machine data:

MD32420 $MA_ JOG_AND_POS_JERK_ENABLE (basic setting of the axial jerk limiting)

You can specify your own axial acceleration and jerk limitation values for manual travel in JOG:

MD32301 $MA_JOG_MAX_ACCEL (maximum axial acceleration for JOG movements)

MD32436 $MA_JOG_MAX_JERK (maximum axial jerk for JOG movements)

Reference:
Function Manual, Basic Functions; Acceleration (B2)

Display

The JOG main screen appears when JOG mode is selected. This main screen contains values relating to position, feedrate, spindle, and tool.

Coordinate systems

In JOG mode, the user has the option to traverse axes in different coordinate systems.

The following coordinate systems are available:

- Basic coordinate system

  Each axis can be traversed manually.
- Workpiece coordinate system

  Only geometry axes can be traversed manually (channel-specific).

Geometry axes

In manual travel, a distinction must be made as to whether the affected axis is to be traversed as a machine axis (axis-specific) or as a geometry axis (channel-specific).

First we will focus on the characteristics of machine axes. Special features relating to the manual traversal of geometry axes are described in "Manual travel of geometry/orientation axes (Page 209)".

Spindle manual travel

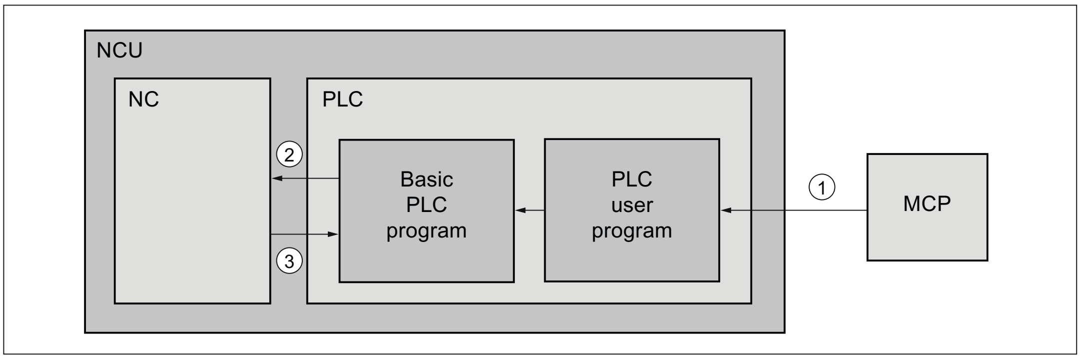
Spindles can also be traversed manually in JOG mode. Essentially, the same conditions apply as for manual travel of axes. Spindles can be traversed in JOG mode using the traversing keys continuously or incrementally, in jog or continuous mode, or using the handwheel. The mode is selected and activated via the axis-/spindle-specific PLC interface as for the axes. The axis-specific machine data also applies to the spindles. Special features relating to the manual traversal of spindles are described in "Spindle manual travel (Page 211)".

3.1.3 Control of manual-travel functions via PLC interface

The manual traversing functions in the NC are activated via the NC/PLC interface from the machine control panel (MCP). To do this, the input signals of the machine control panel from the PLC must be transferred to the input interface of the NC in the NC/PLC interface. In this way, the machine manufacturer can simply adapt the manual traversing functionality to the machine tool through the PLC user program. You can use this, for example, to change the assignment between the direction keys of the machine control panel and the traversing requests to the NC with regard to the machine and geometry axes.

Basic procedure

The following figure shows the basic procedure for the selection of JOG mode from the machine control panel (MCP) to the NC.



① The operator selects, for example, the "Continuous JOG" machine function on the machine control panel for a machine axis.

The input signals of the MCP are transferred cyclically from the basic PLC program in the data blocks of the MCP input interface.

② The PLC user program reads the input signals of the MCP. The input signals can be linked with any other signals in accordance with the current machine or machining situation. Finally, the PLC user program writes the corresponding request signals to the NC in the respective axial NC/PLC interface.

The basic PLC program transfers the request signals in the internal input interface to the NC.

③ After activation of the requested function, the NC writes the feedback in the internal output interface to the PLC. The basic PLC program cyclically transfers the output signals in the respective axial NC/PLC interface

References

Detailed information on the configuration and integration of machine control panels in the PLC user program can be found in the Basic Functions manual:

- SINUMERIK 840D sl: "P3: Basic PLC program for SINUMERIK 840D sl"
- SINUMERIK 828D: "P4: PLC for SINUMERIK 828D"

3.2 Continuous (JOG CONT)

3.2.1 General functionality

Selection

Continuous mode in JOG mode must be selected via the PLC interface:

DB21, ... DBX13.6 (machine function: continuous)

Once the continuous procedure is active, the following interface signal is returned to the PLC:

DB21, … DBX41.6 (active machine function: continuous)

Traversing keys

The plus and minus traversing keys are selected to move the relevant axis in the appropriate direction. If both traversing keys are pressed simultaneously, there is no traversing movement, or, if an axis is in motion, it is stopped.

Note

When the control is switched on, axes can be traversed to the limits of the machine because they have not yet been referenced. Hardware limit switches might be triggered as a result. The software limit switches and the working-area limitation are not operative.

Traversing command plus/minus

As soon as a traverse request for an axis is active (e.g. after selection of a traverse key), one of the following two interface signals is sent to the PLC (depending on selected traverse direction):

DB21, ... DBX40.7 (traversing command plus)

or

DB21, … DBX40.6 (traversing command minus)

3.2.2 Distinction between inching mode continuous mode

Continuous traversing in jog mode can be performed in jogging or in continuous traversing.

Continuous traversing in jog mode

In jog mode, the axis traverses as long as the traversing key is pressed. With the release of the traversing key, the axis is decelerated to standstill. The movement is completed when the parameterized exact stop criterion of the axis is reached.

If the axis reaches a traversing range limit before the traversing key is released (working area limitation, software limit switch, etc.), the axis is stopped at the limit.

Continuous traversing in continuous mode

In continuous mode, traversing of the axis is started by pressing the traversing key. The traversing movement is continued even after the traversing key is released.

> ⚠ WARNING
>
> **Risk of collision**
>
> In continuous mode, several axes can be started by pressing the respective traversing key. Any interlocks must be implemented by the user / machine manufacturer via the PLC user program.

The traversing movement can be interrupted and continued by the operator at any time or aborted. If the axis reaches a traversing range limit before canceling the traversing movement (working area limitation, software limit switch, etc.), the axis is stopped at the limit.

Interrupt traversing movement

The operator can interrupt traversing via the user interface of the machine control panel (MCP) in the following ways:

- Feedrate override = 0%
- Feedrate stop
- NC stop or NC stop axis/spindle

If the cause of the interruption is removed, the axis continues to traverse.

Abort traversing movement

The operator can abort traversing via the user interface of the machine control panel (MCP) in the following ways:

- Pressing the same traversing key again
- Pressing the traversing key for the opposite direction
- RESET
- Deselection of the JOG mode by changing the operating mode to AUTOMATIC or MDI

The traversing movement is aborted from the control when:

- An active traversing range limit is reached (working area limitation, software limit switch, etc.)

> ⚠ CAUTION
>
> **Traversing range limit inactive**
>
> Software limit switches and working-area limitations are only active after referencing of the axis.

- An alarm occurs with cancellation of the traversing movements

Parameter assignment

The selection of jog or continuous mode is performed NC-specifically for all axes via the setting data:

SD41050 $SN_JOG_CONT_MODE_LEVELTRIGGRD (jog/continuous mode for JOG continuous)

3.2.3 Supplementary conditions

Indexing axis

An indexing axis always stops at an indexing position both in jog mode and in continuous mode. In jog mode, the indexing axis traverses to the next indexing position in the direction of travel, for example, when the traversing key is released (see Section "T1: Indexing axes (Page 771)").

3.3 Incremental (JOG INC)

3.3.1 General functionality

Function

With incremental traversing, the operator specifies the number of increments to be traversed by the axis via the machine control panel.

In addition to five fixed increment sizes (default setting: INC1, INC10, INC100, INC1000 and INC10000), a variable increment size (INCvar) that can be set via the setting data is also available.

Parameter assignment

Fixed increments

The parameter assignment of the fixed increment sizes is performed via NC-specific machine data:

MD11330 $MN_JOG_INCR_SIZE_TAB[1 ... 5] = <number of increments 1 ... 5>

Variable increment

The parameter assignment of the variable increment size is performed via NC-specific setting data:

SD41010 $SN_JOG_VAR_INCR_SIZE = <number of increments>

Distance evaluation of one increment

The distance evaluation of one increment for fixed and variable increment sizes is performed via the axis-specific machine data:

MD31090 $MA_JOG_INCR_WEIGHT = <distance>

Note

Reversal of the direction evaluation

The input of a negative value causes a reversal of the direction evaluation of the traversing keys or the handwheel direction of rotation.

NC/PLC interface

Selection, axial

DB31, ... DBX5.0 - 5.5 (machine function: INC1 to INCvar)

Feedback, axial

DB31, ... DBX65.0 - 65.5 (active machine function: INC1 to INCvar)

3.3.2 Distinction between jogging mode and continuous mode

Analogous to the continuous traversing in JOG mode, incremental traversing can also be performed in jogging or in continuous traversing.

Incremental travel in jogging mode

Function

If the traversing key for the required direction (e.g. +) is pressed, the axis begins to traverse the increment that has been set. If the traversing key is released before the increment has been fully traversed, the movement is interrupted and the axis stops. If the same traversing key is pressed again, the axis traverses the remaining distance until it is zero. Up to this point, the movement can still be interrupted by releasing the traversing key.

Pressing the traversing key for the opposite direction does not have any effect, unless the increment has been fully traversed or the movement has been aborted.

Abort traversing movement

If the increment should not travel to the end, the traversing movement can be interrupted as follows:

- RESET
- DB31, ... DBX2.2 (delete distance-to-go)

Incremental travel in continuous mode

Function

The axis traverses the entire set increment when the traversing key is pressed (first rising edge). If the same traversing key is pressed again (second rising edge) before the axis has finished traversing the increment, the traversing movement is aborted, i.e. not completed.

Interrupt traversing movement

Behavior as for continuous travel.

Abort traversing movement

The traversing movement can be stopped and aborted by means of the following operations or monitoring functions:

- Pressing the same traversing key again
- Pressing the traversing key for the opposite direction
- RESET
- DB31, ... DBX2.2 (delete distance-to-go)

- When the first valid limit is reached

> ⚠ CAUTION
>
> **Traversing range limit inactive**
>
> Software limit switches and working-area limitations are only activated after reference point approach.

- Deselection or change of the current increment (e.g. change from INC100 to INC10)
- When faults occur (e.g. on cancellation of the servo enable)

> ⚠ WARNING
>
> **Risk of collision**
>
> If "continuous" mode is selected, several axes can by started by pressing and releasing the relevant direction key. Any interlocks must be implemented via the PLC!

Note

While an axis is moving, a change of mode from JOG to AUTOMATIC or MDI is not permitted within the control.

Parameter assignment

The selection of jog or continuous mode is performed NC-specifically for all axes via the machine data:

MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD (INC and REF in jog mode)

Jogging mode is the default setting.

3.3.3 Supplementary conditions

Indexing axis

Irrespective of the set increment value, an indexing axis traverses to the next indexing position in the direction of travel after pressing the traversing key (see Section "T1: Indexing axes (Page 771)").

3.4 Handwheel travel in JOG

3.4.1 Function

The electronic handwheels (accessories) can be used to simultaneously traverse selected axes manually. The weighting of the handwheel graduations is dependent on the increment-size weighting. Where coordinate offset or rotation is selected, manual travel can even be performed in the transformed workpiece coordinate system.

Select

JOG mode must be active. The user must also set the increment INC1, INC10, etc., which applies to handwheel travel. As with incremental travel, the required machine function must be set at the PLC interface accordingly.

Traversing

When the electronic handwheel is turned, the associated axis is traversed either in the positive or negative direction depending on the direction of rotation.

Note

If the axis is already being moved using the traversing keys, the handwheel cannot be used.

Traversing distance

The traversing distance produced by rotating the handwheel is dependent on the following factors:

- Number of handwheel pulses received at the interface
- Active increment (machine function INC1, INC10, INC100, ... INCvar)
- Pulse evaluation of the handwheel:

  MD11320 $MN_HANDWH_IMP_PER_LATCH (handwheel pulses per detent position)

  See Section "Parameter assignment (Page 161)".
- Distance of an increment:

  MD31090 $MA_JOG_INCR_WEIGHT (evaluation of an increment for INC/handwheel)

  See Section "Parameter assignment (Page 161)".

Handwheel connection

Up to six handwheels can be connected simultaneously. This means that up to six axes can be traversed by handwheel simultaneously.

Representation of the handwheel number in the NC/PLC interface signals

Depending on the parameter assignment of MD11324, the representation of the handwheel number in the NC/PLC interface signals is **bit-coded** (three handwheels can be represented) or **binary-coded** (six handwheels can be represented).

See "Parameter assignment (Page 161)".

Handwheel assignment

It can be set as to which axis is moved by turning the handwheel:

- via the PLC user interface or
- via the user interface (HMI).

The assignment is linked to the NC/PLC interface through the PLC user program. In this way, several axes can be assigned to one handwheel simultaneously.

Setting via the PLC user interface

The assignment is made using one of the following interface signals:

- Machine axes:
  - DB31, ... DBX4.0-2 (activate handwheel <n> (1, 2, 3))
- Geometry axes:
  - DB21, ... DBX12.0-2 (geometry axis 1: Activate handwheel <n>)
  - DB21, … DBX16.0-2 (geometry axis 2: Activate handwheel <n>)
  - DB21, … DBX20.0-2 (geometry axis 3: Activate handwheel <n>)
- Orientation axes:
  - DB21, … DBX320.0-2 (orientation axis 1: Activate handwheel <n>)
  - DB21, … DBX324.0-2 (orientation axis 2: Activate handwheel <n>)
  - DB21, … DBX328.0-2 (orientation axis 3: Activate handwheel <n>)

Setting via the user interface (HMI).

Pressing the "Handwheel" softkey in the JOG-mode basic menu displays the "Handwheel" window. Here, every handwheel can be assigned an axis and the handwheel can be enabled or disabled.

Note

The handwheel assignment is **not** possible via the user interface (HMI) for more than three connected handwheels and a binary-coded representation of the handwheel number in the NC/PLC interface signals.

Handwheel selection by HMI

A separate user interface is provided between the HMI and PLC to allow activation of the handwheel from the user interface. This interface supplied by the basic PLC program for handwheels 1, 2 and 3 contains the following information:

- Assigned to the handwheel:
  - Axis number (if during the handwheel selection a machine axis was selected):

    DB10 DBX100.0-4 (axis number for handwheel 1)

    DB10 DBX101.0-4 (axis number for handwheel 2)

    DB10 DBX102.0-4 (axis number for handwheel 3)
  - Channel number (if during the handwheel selection a geometry axis was selected):

    DB10 DBX97.0-3 (channel number for handwheel 1)

    DB10 DBX98.0-3 (channel number for handwheel 2)

    DB10 DBX99.0-3 (channel number for handwheel 3)
- Additional information on the machine or geometry axis:

  DB10 DBX100.7 (handwheel 1: Machine axis)

  DB10 DBX101.7 (handwheel 2: Machine axis)

  DB10 DBX102.7 (handwheel 3: Machine axis)
- The information that the handwheel is enabled or disabled:

  DB10 DBX100.6 (handwheel 1 selected)

  DB10 DBX101.6 (handwheel 2 selected)

  DB10 DBX102.6 (handwheel 3 selected)

For the specified axis, the basic PLC program sets the associated interface signal either to "0" (disable) or to "1" (enable):

- Machine axes:
  - DB31, ... DBX4.0-2 (activate handwheel <n>)
- Geometry axes:
  - DB21, ... DBX12.0-2 (geometry axis 1: Activate handwheel <n>)
  - DB21, … DBX16.0-2 (geometry axis 2: Activate handwheel <n>)
  - DB21, … DBX20.0-2 (geometry axis 3: Activate handwheel <n>)

Note

Orientation axes can only be activated via the associated PLC user interface signals:

- DB21, … DBX320.0-2 (orientation axis 1: Activate handwheel <n>)
- DB21, … DBX324.0-2 (orientation axis 2: Activate handwheel <n>)
- DB21, … DBX328.0-2 (orientation axis 3: Activate handwheel <n>)

Travel request

The following NC/PLC interface signal informs the PLC that an axis wants to travel or is travelling:

- Machine axes:
  – DB31, ... DBX64.4 (minus travel request) or
  – DB31, ... DBX64.5 (plus travel request)
- Geometry axis 1:
  – DB21, … DBX40.4 (geometry axis 1: Minus travel request) or
  – DB21, … DBX40.5 (geometry axis 1: Plus travel request)
- Geometry axis 2:
  – DB21, … DBX46.4 (geometry axis 2: Minus travel request) or
  – DB21, … DBX46.5 (geometry axis 2: Plus travel request)
- Geometry axis 3:
  – DB21, … DBX52.4 (geometry axis 3: Minus travel request) or
  – DB21, … DBX52.5 (geometry axis 3: Plus travel request)
- Orientation axis 1:
  – DB21, … DBX332.4 (orientation axis 1: Minus travel request) or
  – DB21, … DBX332.5 (orientation axis 1: Plus travel request)
- Orientation axis 2:
  – DB21, … DBX336.4 (orientation axis 2: Minus travel request) or
  – DB21, … DBX336.5 (orientation axis 2: Plus travel request)
- Orientation axis 3:
  – DB21, … DBX340.4 (orientation axis 3: Minus travel request) or
  – DB21, … DBX340.5 (orientation axis 3: Plus travel request)

Note

A travel request is the sum of all sub-movements, i.e. the component from couplings and offset values is also taken into account.

For the method of operation of the "Travel request" function, see Section "Travel request (Page 166)".

Travel command

Depending on the setting in machine data MD11324 $MN_HANDWH_VDI_REPRESENTATION (see Section "Parameter assignment (Page 161)"), the following interface signal is output to the PC already when a travel request is present or not until the axis moves:

- Machine axes:
  - DB31, ... DBX64.6 (minus travel command) or
  - DB31, ... DBX64.7 (plus travel command)
- Geometry axis 1:
  - DB21, … DBX40.6 (geometry axis 1: Minus travel command) or
  - DB21, … DBX40.7 (geometry axis 1: Plus travel command)
- Geometry axis 2:
  - DB21, … DBX46.6 (geometry axis 2: Minus travel command) or
  - DB21, … DBX46.7 (geometry axis 2: Plus travel command)
- Geometry axis 3:
  - DB21, … DBX52.6 (geometry axis 3: Minus travel command) or
  - DB21, … DBX52.7 (geometry axis 3: Plus travel command)
- Orientation axis 1:
  - DB21, … DBX332.6 (orientation axis 1: Minus travel command) or
  - DB21, … DBX332.7 (orientation axis 1: Plus travel command)
- Orientation axis 2:
  - DB21, … DBX336.6 (orientation axis 2: Minus travel command) or
  - DB21, … DBX336.7 (orientation axis 2: Plus travel command)
- Orientation axis 3:
  - DB21, … DBX340.6 (orientation axis 3: Minus travel command) or
  - DB21, … DBX340.7 (orientation axis 3: Plus travel command)

Invert handwheel direction of rotation

The handwheel direction of rotation can be inverted, if the direction of movement of the handwheel does not match the expected direction of motion of the axis. The adaptation can be especially necessary, if a handwheel (HT2, HT8) can be assigned to various axes.

In addition to configuring the particular MD, handwheel direction of rotation inversion can be activated by setting the IS "Invert the handwheel direction of rotation" belonging to the particular axis:

- Machine axes:
  - DB31, ... DBX7.0 (invert handwheel direction of rotation)
- Geometry axes:
  - DB21, ... DBX15.0 (geometry axis 1: Invert handwheel direction of rotation)
  - DB21, ... DBX19.0 (geometry axis 2: Invert handwheel direction of rotation)
  - DB21, ... DBX23.0 (geometry axis 3: Invert handwheel direction of rotation)
- Orientation axes:
  - DB21, … DBX323.0 (orientation axis 1: Invert handwheel direction of rotation)
  - DB21, … DBX327.0 (orientation axis 2: Invert handwheel direction of rotation)
  - DB21, … DBX331.0 (orientation axis 3: Invert handwheel direction of rotation)
- Contour handwheel:
  - DB21, ... DBX31.5 (invert contour handwheel direction of rotation)

Note

The inversion signal should be set in the PLC user program at the same time as the handwheel selection (IS "Activate handwheel").

The acknowledgement that the handwheel direction of rotation has been inverted by the NC is realized for each axis using the IS "Handwheel direction of rotation inversion active":

- Machine axes:
  - DB31, ... DBX67.0 (handwheel direction of rotation inversion active)
- Geometry axes:
  - DB21, ... DBX43.0 (geometry axis 1: Handwheel direction of rotation inversion active)
  - DB21, ... DBX49.0 (geometry axis 2: Handwheel direction of rotation inversion active)
  - DB21, ... DBX55.0 (geometry axis 3: Handwheel direction of rotation inversion active)
- Orientation axes:
  - DB21, ... DBX335.0 (orientation axis 1: Handwheel direction of rotation inversion active)
  - DB21, ... DBX339.0 (orientation axis 2: Handwheel direction of rotation inversion active)
  - DB21, ... DBX343.0 (orientation axis 3: Handwheel direction of rotation inversion active)
- Contour handwheel:
  - DB21, ... DBX39.5 (contour handwheel direction of rotation inversion active)

Note

It is only permissible to change the inversion signal at standstill. If the change is made while motion setpoints are being output by the interpolator, then the signal change is rejected and an alarm is output; further, motion is stopped taking into account the actual acceleration value.

Abort/interruption of the traversing motion

The following NC/PLC interface signals abort the traversing motion. The setpoint/actual-value difference is deleted.

NC/PLC interface signal	Scope	Effect	
DB21, ... DBX7.7 (NC reset)	Geometry axis / machine axis	Abort	
DB21, ... DBX12.0-2 (geometry axis 1: Activate handwheel <n>) DB21, … DBX16.0-2 (geometry axis 2: Activate handwheel <n>) DB21, … DBX20.0-2 (geometry axis 3: Activate handwheel <n>)	Geometry axis	1 → 0	Abort
DB21, ... DBX6.2 (delete distance-to-go)	Geometry axis / machine axis	0 → 1	Abort
DB31, ... DBX2.2 (delete distance-to-go / spindle reset)	Geometry axis / machine axis	0 → 1	Abort
DB31, ... DBX4.2 (activate handwheel <n>)	Machine axis	1 → 0	Abort
DB31, ... DBX1.3 (axis/spindle disable)	Geometry axis / machine axis	0 → 1	Abort
DB31, ... DBX1.5 (position measuring system 1)	Geometry axis / machine axis	0 → 1	Abort
DB31, ... DBX1.6 (position measuring system 2)	Geometry axis / machine axis	0 → 1	Abort

The effect of the handwheel travel (abort or interruption of the traversing motion) can be set for other NC/PLC interface signals (stop signals) (see Section "Parameter assignment (Page 161)").

NC STOP only interrupts the traversing movement. Any setpoint/actual-value difference is retained. The distance-to-go is then traversed using NC START.

Limitations

The limitations are also active when traversing with the handwheel.

For further information, see Section "Monitoring functions (Page 213)".

3.4.2 Parameter assignment

Distance or velocity specification

Either the distance or the velocity can be entered via the handwheel:

- Distance specification (default setting)

  The distance specified by the handwheel is traversed and No pulses are lost.

  Limiting the velocity to the maximum permissible value causes the axes to overtravel.
- Velocity specification

  The handwheel only defines the traverse velocity. As soon as the handwheel stops, the axes stop too. Motion is braked immediately if no pulses are supplied from the handwheel in one IPO cycle, thus preventing overtravel by the axes. The handwheel pulses do not supply a path default.

The input mode is set with machine data:

MD11346 $MN_HANDWH_TRUE_DISTANCE (handwheel distance or velocity specification)

Pulse evaluation of the handwheel

The number of pulses that are to be generated per handwheel detent position must be specified for each handwheel:

MD11320 $MN_HANDWH_IMP_PER_LATCH [<n>] (handwheel pulses per detent position)

Note

Input of a negative value results in a reversal of the handwheel direction of rotation.

Increment size

The number of increments to be traversed by the axis per handwheel pulse is determined by the selected increment size.

In addition to five fixed increment sizes (default setting: INC1, INC10, INC100, INC1000 and INC10000), a variable increment size (INCvar) that can be set via the setting data is also available.

Fixed increments

The parameter assignment of the fixed increment sizes is performed via NC-specific machine data:

MD11330 $MN_JOG_INCR_SIZE_TAB[1 ... 5] = <number of increments 1 ... 5>

Variable increment

The parameter assignment of the variable increment size is performed via NC-specific setting data:

SD41010 $SN_JOG_VAR_INCR_SIZE = <number of increments>

Distance evaluation of one increment

The distance evaluation of one increment for fixed and variable increment sizes is performed via the axis-specific machine data:

MD31090 $MA_JOG_INCR_WEIGHT = <distance>

Note

Input of a negative value results in a reversal of the handwheel direction of rotation.

Limitation of the increment size

The machine operator can limit the size of the selected increment:

- For machine axes, using the axis-specific machine data:

  MD32080 $MA_HANDWH_MAX_INCR_SIZE (limiting the selected increment)
- For geometry axes, using the channel-specific machine data:

  MD20620 $MC_HANDWH_GEOAX_MAX_INCR_SIZE (limitation of handwheel increment for geometry axes)
- For orientation axes, using the channel-specific machine data:

  MD20621 $MC_HANDWH_ORIAX_MAX_INCR_SIZE (limitation of handwheel increment for orientation axes)

Representation of the handwheel number in the NC/PLC interface signals

The representation of the handwheel number in the NC/PLC interface signals is defined using machine data:

MD11324 $MN_HANDWH_VDI_REPRESENTATION

Value	Meaning
0	**Bit-coded** representation (**basic setting**) → Three handwheels can be represented.
1	**Binary-coded** representation → Six handwheels can be represented.

Output of the NC/PLC interface signals "Plus travel command" / "Minus travel command"

The output behavior of the NC/PLC interface signals "Plus travel command" / "Minus travel command" is specified with the machine data:

MD17900 $MN_VDI_FUNCTION_MASK

Value	Meaning
0	The NC/PLC interface signals "Plus travel command" / "Minus travel command" are already output when a travel request is active (default setting).
1	The NC/PLC interface signals "Plus travel command" / "Minus travel command" are only output when the axis actually moves (i.e. if setpoints are output at the servo).

Velocity

In handwheel travel the following axis velocities, effective during JOG mode, are used:

- SD41110 $SN_JOG_SET_VELO (axis velocity for JOG)
- SD41130 $SN_JOG_ROT_AX_SET_VELO (axis velocity for rotary axes for JOG mode)
- MD32020 $MA_JOG_VELO (conventional axis velocity)

Because of the limited feedrate, the axis is not able to follow the handwheel rotation synchronously, especially in the case of a large pulse weighting, and therefore overtravels.

Acceleration

As for manual travel (see "General characteristics when traversing in the JOG mode (Page 144) ").

Movement in the opposite direction

The behavior at a reversal of the traversing direction (by turning the handwheel in the opposite direction) can be set in the machine data:

MD11310 $MN_HANDWH_REVERSE (threshold for direction change handwheel)

Value	Meaning
= 0	If the handwheel is moved in the opposite direction, the resulting distance is computed and the calculated end point is approached as fast as possible. If this end point is located before the point where the moving axis can decelerate in the current direction of travel, the unit is decelerated and the end point is approached by moving in the opposite direction. If this is not the case, the newly calculated end point is approached immediately.
> 0	If the handwheel is moved in the opposite direction by at least the number of pulses indicated in the machine data, the axis is decelerated as fast as possible and all pulses received until the end of the interpolation are ignored. This means that another movement takes place only after the axis reaches standstill (setpoint side).

Behavior at software limit switches, working-area limitation

When axes are traversed in JOG mode, they can traverse only up to the first active limitation before the corresponding alarm is output.

Depending on the setting in the machine data:
MD11310 $MN_HANDWH_REVERSE (threshold for direction change, handwheel)
the behavior is then as follows (as long as the axis has still not arrived at the end point from the setpoint side):

- The distance resulting from the handwheel pulses forms a fictitious end point which is used for subsequent calculations.

  If this fictitious end point is, for example, 10 mm behind the limit, these 10 mm must be traversed in the opposite direction before the axis traverses again. If a movement in the opposite direction is to be performed immediately after a limitation is reached, the fictitious distance-to-go can be deleted via delete distance-to-go or deselection of the handwheel assignment.
- All handwheel pulses leading to an end point behind the limitation are ignored. Any movement of the handwheel in the opposite direction leads to an immediate movement in the opposite direction, i.e. away from the limit.

Feedrate behavior

In JOG mode, the feedrate behavior of the axis/spindle also depends on the setting data:

SD41100 $SN_JOG_REV_IS_ACTIVE (JOG: Revolutional/linear feedrate)

SD41100 $SN_JOG_REV_IS_ACTIVE	
Active	An axis/spindle is always traversed with revolutional feedrate MD32050 $MA_JOG_REV_VELO (revolutional feedrate for JOG) or MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feedrate for JOG with rapid traverse override) depending on the master spindle.
Not active	The behavior of the axis/spindle depends on the setting data: SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (revolutional feedrate for positioning axes/spindles)
	The behavior of a geometry axis on which a frame with rotation acts, depends on the channel-specific setting data: SD42600 $SC_JOG_FEED_PER_REV_SOURCE (control of the revolutional feedrate in JOG)

Effect of the NC/PLC interface stop signals

The effect of the NC/PLC interface stop signals on the handwheel travel (abort or interruption of the traversing motion) can be set via the machine data:

MD20624 $MC_HANDWH_CHAN_STOP_COND (definition of the handwheel travel behavior, channel-specific)

MD32084 $MA_HANDWH_STOP_COND (handwheel travel behavior)

NC/PLC interface signal	Scope	MD20624 $MC_HANDWH_CHAN_STOP_COND	
		Bit == 0	Bit == 1
DB11 DBX0.5 (mode group stop)	Geometry axis / machine axis	Interruption until NC start	Abort
DB11 DBX0.6 (mode group stop, axes plus spindle)	Geometry axis / machine axis	Interruption until NC start	Abort
DB21, ... DBX7.3 (NC Stop)	Geometry axis / machine axis	Interruption until NC start	Abort
DB21, ... DBX7.4 (NC stop, axes plus spindle)	Geometry axis / machine axis	Interruption until NC start	Abort
DB21, ... DBX6.0 (feedrate disable)	Geometry axis / machine axis	Interruption	Abort
DB21, ... DBB4 (feedrate override) DB31, ... DBX4.5 (rapid traverse override)	Geometry axes	Override == 0: Interruption if rapid traverse override is not active	Override == 0: Abort if rapid traverse override is not active
DB21, ... DBB5 (rapid traverse override) DB31, ... DBX4.5 (rapid traverse override)	Geometry axes	Rapid traverse override == 0: Interruption if rapid traverse override is active	Rapid traverse override == 0: Abort if rapid traverse override is active
DB21, ... DBX12/16/20.3 (geometry axis 1/2/3: Feedrate stop)	Geometry axis	Interruption	Abort

NC/PLC interface signal	Scope	MD32084 $MC_HANDWH_CHAN_STOP_COND	
		Bit == 0	Bit == 1
DB31, ... DBB0 (axial feedrate override)	Machine axis, not spindle	Override == 0: Interruption	Override == 0: Abort
DB31, ... DBB19 (spindle override)	Spindle	Override == 0: Interruption	Override == 0: Abort
DB31, ... DBX4.3 (feedrate stop / spindle stop)	Geometry axis / machine axis	Interruption	Abort
DB31, ... DBX2.3 (clamping in progress)	Geometry axis / machine axis	No effect	Abort
DB31, ... DBX2.1 (control system enable)	Geometry axis / machine axis	Interruption	Abort
DB31, ... DBX21.7 (pulse enable)	Geometry axis / machine axis	Interruption	Abort

Note

Removal of the "Controller enable" and "Pulse enable" during traversing results in a rapid stop of the axis and a reset alarm.

Interruption of a traversing motion

When a stop command is issued, the distance-to-go is saved and the handwheel pulses are collected. When the stop condition no longer applies, the resulting distance is traversed.

Abort of a traversing motion

When a stop command is issued, the distance-to-go is deleted and the handwheel pulses are ignored (i.e. not collected) until the stop condition no longer applies.

3.4.3 Travel request

The following examples are intended to illustrate the method of operation of the "Travel request" NC/PLC interface signal.

Example 1: Handwheel travel with distance specification, stop condition is not an abort criterion

If a stop condition that is present is not an abort criterion (see MD32084 $MA_HANDWH_STOP_COND or MD20624 $MC_HANDWH_CHAN_STOP_COND) during handwheel travel with distance specification (MD11346 $MN_HANDWH_TRUE_DISTANCE == 1 or == 3), then the output of the NC/PLC interface signals "Travel request" and "Travel command" corresponds to the behavior in the two figures below.

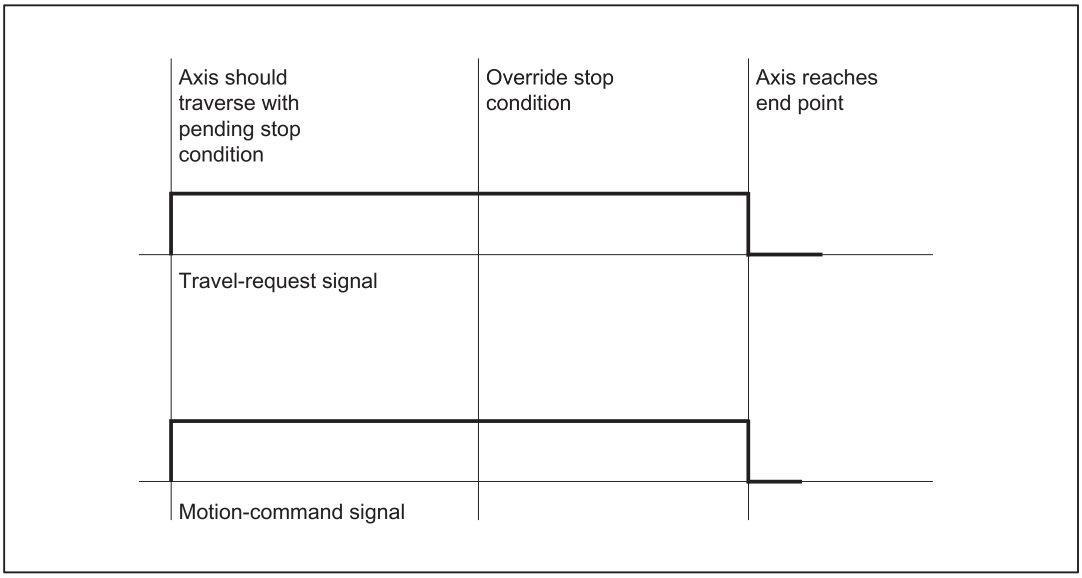


Figure 3-1 Signal-time diagram:
Handwheel travel with distance specification, stop condition is not an abort criterion
MD17900 $MN_VDI_FUNCTION_MASK bit 0 = 0

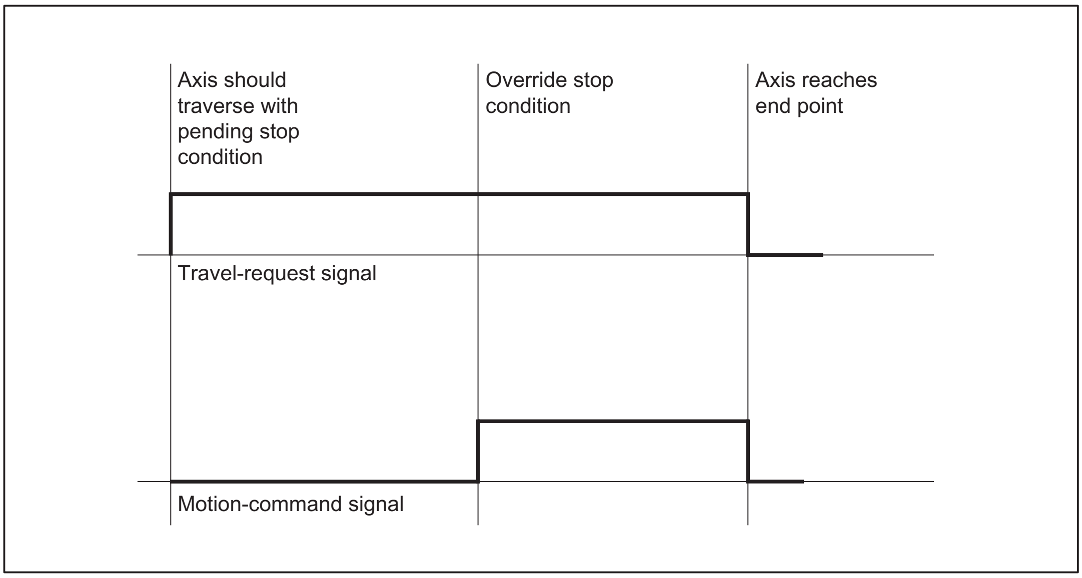


Figure 3-2 Signal-time diagram:
Handwheel travel with distance specification, stop condition is not an abort criterion
MD17900 $MN_VDI_FUNCTION_MASK bit 0 = 1

Example 2: Handwheel travel, stop condition is an abort criterion

If a stop condition is selected as an abort criterion via machine data
MD32084 $MA_HANDWH_STOP_COND or
MD20624 $MC_HANDWH_CHAN_STOP_COND during handwheel travel, **no travel command** is output, **but** the corresponding **travel request** is output.

When the stop condition is removed, the corresponding "Travel request" NC/PLC interface signal is reset, as an abort is present. The stop condition is no longer active, but the axis cannot be traversed as the stop condition has caused an abort.

In addition, either the distance specification (MD11346 $MN_HANDWH_TRUE_DISTANCE == 1 or == 3) is active or the handwheel is moved continuously, i.e. it provides pulses.

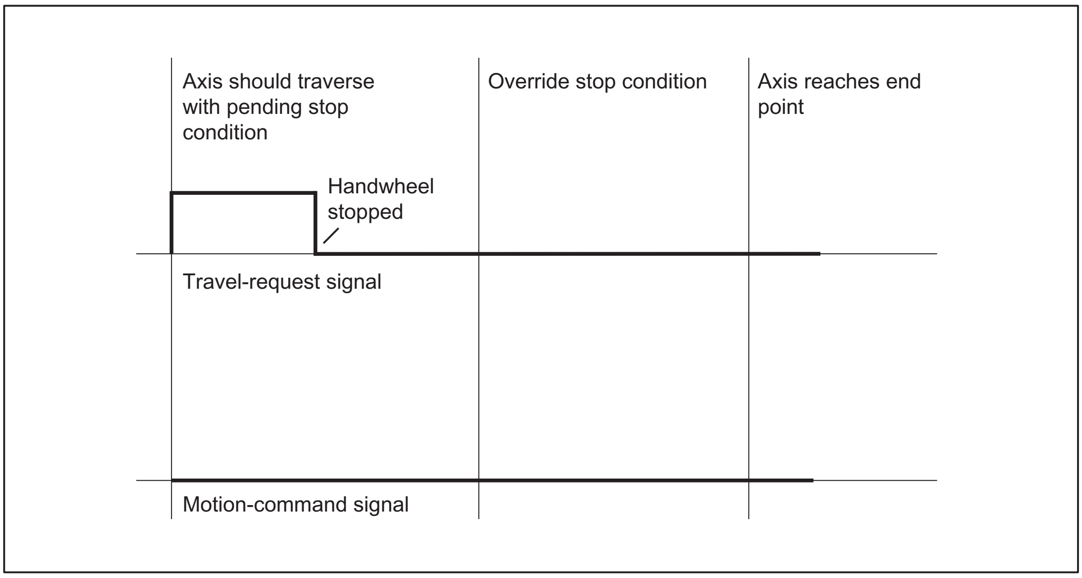


Figure 3-3 Signal-time diagram:
Handwheel travel, stop condition is an abort criterion

If a stop condition is activated during the handwheel travel, the motion is aborted and the "Travel request" and "Travel command" NC/PLC interface signals are reset.

Example 3: Handwheel travel with velocity specification, stop condition is an abort criterion

If the handwheel is no longer moved for velocity specification (MD11346 $MN_HANDWH_TRUE_DISTANCE == 0 or == 2), the "Travel request" NC/PLC interface signal is reset.

The "Travel request" PLC signal is also reset when the handwheel is deselected.

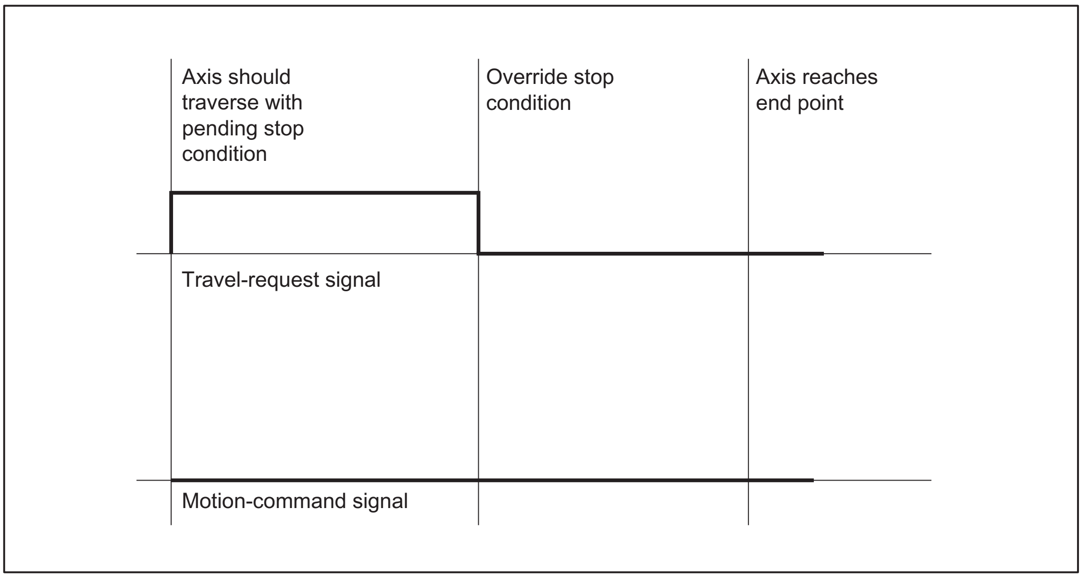


Figure 3-4 Signal-time diagram:
Handwheel travel with velocity specification, stop condition is an abort criterion

Supplementary conditions

NC stop

With NC stop present, no travel command and, therefore, no travel request is output. There is an exception with DRF travel: If DRF travel is permitted in the NC stop state via machine data MD20624 $MC_HANDWH_CHAN_STOP_COND (bit 13 == 1), the behavior corresponds to that of handwheel travel.

3.4.4 Double use of the handwheel

Alarm 14320

The double use of a handwheel for DRF and velocity or distance overlay, including contour handwheel, is suppressed and is displayed using the self-clearing alarm 14320 (Handwheel %1 used twice (%2) in channel %3 axis %4), if, different influences can act on an axis as a result of the handwheel.

This means that an overlaid movement can only be executed when no DRF offset (triggered by the same handwheel) is active for the axes in the basic coordinate system that are involved in the movement, i.e., the DRF movement must have been terminated.

If an overlaid movement has been started, no DRF offset can be started for any of the axes involved that are supplied by the same handwheel. Such a DRF movement is only possible when the movement with overlay has reached its end point or has been aborted by delete distance-to-go or RESET.

If the handwheel override and DRF offset are to be active simultaneously, this is possible with activation of two separate handwheels.

Example: Path override

Assumption:
Channel 1 and geometry axis X correspond to machine axis 3 and geometry axis Y corresponds to machine axis 5 and handwheel 2 is selected for the first geometry axis.

If block `X10 Y10 FD=0` is processed in the main run, neither machine axis 3 nor machine axis 5 can be traversed with DRF via handwheel 2. If handwheel 2 is assigned to machine axis 3 while the channel-specific DRF signal is active, then alarm 14320 (Handwheel 2 used twice (8) in channel 1 axis X) is signaled.

If machine axis 3 or machine axis 5 is traversed with DRF using the 2nd handwheel, then motion `X10 Y10 FD=0` cannot be executed and alarm 14320 (handwheel 2 used twice (3) in channel 1 axis X) or alarm 14320 (handwheel 2 used twice (3) in channel 1 axis Y) is signaled.

Example: Velocity override of positioning axis

Assumption:
Channel 1: Channel axis A corresponds to machine axis 4 and handwheel 1 is assigned to this axis.

If block `POS[A]=100 FDA[A]=0` is processed in the main run, machine axis 4 cannot be traversed with DRF. This means that if the channel-specific DRF signal is active, alarm 14320 (Handwheel 1 used twice (6) in channel 1 axis A) is signaled.

If machine axis 4 is traversed with DRF, then no `POS[A]=100 FDA[A]=0` movement can be executed while a DRF movement is being performed. Alarm 14320 (Handwheel 1 used twice (1) in channel 1 axis A) is signaled.

Example: Distance overlay PLC axis (840D sl)

Assumption:
Channel 1: Handwheel 2 is assigned to machine axis 4.

If an axis movement with path override of the 4th machine axis triggered by FC18 is processed in the main run, machine axis 4 cannot be traversed with DRF. This means that if the channel-specific DRF signal is active, alarm 14320 (Handwheel 2 used twice (9) in channel 1 axis A) is signaled.

If machine axis 4 is traversed with DRF, then no axis movement with path override triggered by FC18 can be executed while a DRF movement is being performed. Alarm 14320 (Handwheel 2 used twice (4) in channel 1 axis A) is signaled.

3.5 Handwheel override in automatic mode

3.5.1 General functionality

Function

With this function it is possible to traverse axes or to change their velocities directly with the handwheel in automatic mode (Automatic, MDA).

The handwheel override is activated in the NC part program using the NC language elements `FD` (for path axes) and `FDA` (for positioning axes) and is non-modal.

With positioning axes, it is possible to activate the handwheel override modally using traverse command `POSA`. When the programmed target position is reached, the handwheel override becomes inactive again.

Additional axes can be traversed simultaneously or using interpolation in the same NC block.

The concurrent-positioning-axes function can also be activated by the PLC user program.

Distinction

Depending on the programmed feedrate, a distinction is made between the following for handwheel override:

- **Path definition**

  Axis feedrate = 0 (FDA = 0)

- **Velocity override**

  Axis feedrate > 0 (FD or FDA > 0)

The table below shows which axis types can be influenced by the "handwheel override in automatic mode" function.

Axes that can be influenced by the "handwheel override in automatic mode" function		
Axis type	**Velocity override**	**Path definition**
Positioning axis	FDA[AXi] > 0 ; acts axially	FDA[AXi] = 0
Concurrent positioning axis	Parameter "Handwheel override active" = 1 and axis feedrate > 0 from FC18	Parameter "Handwheel override active" = 1 and axis feedrate = 0 from FC18
Path axis	FD > 0 ; acts on path velocity	Not possible

Path definition

With axis feedrate = 0 (e.g. `FDA[AXi] = 0`), the traversing movement of the positioning axis towards the programmed target position is controlled entirely by the user rotating the assigned handwheel.

The direction in which the handwheel is turned determines the traversing direction of the axis. The programmed target position cannot be exceeded during handwheel override. The axis can also be moved toward the programmed target position from the opposite direction, whereby the movement is only restricted by the axial position limitations.

A block change occurs when:

- The axis has reached the programmed target position

or

- The distance-to-go is deleted by axial interface signal
  DB31, ... DBX2.2 (delete distance-to-go).

From this moment on, the path default is deactivated and any further handwheel pulses have no effect.

After this, incrementally programmed positions refer to the point of interruption and not to the last programmed position.

Velocity override

With regard to the velocity override, a distinction is made between axis feedrate and path feedrate.

- **Axis-velocity override** (FDA[AXi] > 0):

  The positioning axis is moved to the target position at the programmed axial feedrate. Using the assigned handwheel, it is possible to increase the axis velocity or to reduce it to a minimum of zero depending on the direction of rotation. The resulting axis feedrate is limited by the maximum velocity. However, the axis cannot be traversed in the opposite direction to that programmed.

  The block change is performed as soon as the axis reaches the programmed target position. This causes the velocity override to be deactivated automatically and any further handwheel pulses have no effect.

  Similarly, this also applies to concurrent positioning axes, where the target position and the velocity are defined by the PLC.

- **Path-velocity override** (FD > 0):

  The path axes programmed in the NC block traverse to the target position at the programmed feedrate. If the velocity override is active, the programmed path velocity is overridden by the velocity generated with the **handwheel of the 1st geometry axis**. The block change is performed as soon as the programmed target position is reached.

  The path velocity is increased or reduced to a minimum of zero depending on the direction of rotation of the handwheel. However, it is not possible to reverse the direction of movement with handwheel override.

Application example

The "Handwheel override in automatic mode" function is frequently used on grinding machines. For example, the user can position the reciprocating grinding wheel on the workpiece using the handwheel (path default). After scratching, the traversing movement is terminated and the block change is initiated (by activating DB31, ... DBX2.2 (delete distance-to-go)).

Preconditions

In order to activate "Handwheel override in automatic mode", the following requirements must have been met:

- A handwheel must be assigned to the axis in question.
- Pulse weighting exists for the assigned handwheel.

Handwheel assignment

The assignment of the connected handwheels to the axes is analogous to the "Handwheel travel in JOG (Page 154)" via the user interface or via the PLC user interface with one of the following interface signals:

- Machine axes:
  - DB31, ... DBX4.0-2 (activate handwheel (1, 2, 3))
- Geometry axes:
  - DB21, ... DBX12.0-2 (geometry axis 1: Activate handwheel (1, 2, 3))
  - DB21, … DBX16.0-2 (geometry axis 2: Activate handwheel (1, 2, 3))
  - DB21, … DBX20.0-2 (geometry axis 3: Activate handwheel (1, 2, 3))

If handwheel override is programmed for an axis to which no handwheel is assigned, a distinction is made between the following cases:

- **For velocity override:**

  The axes traverse at the programmed velocity.
  A self-acknowledging alarm is output (without response).
- **For path definition:**

  No traversing movement is performed because the velocity is zero.
  A self-acknowledging alarm is output (without response).

Note

When the velocity override is applied to path axes, only the **handwheel of the 1st geometry axis** acts on the path velocity.

Handwheel weighting

The traverse path of the axis that is generated by rotating the handwheel by one detent position is dependent on several factors (see Section "Handwheel travel in JOG (Page 154)"):

- Selected increment size:

  MD11330 $MN_JOG_INCR_SIZE_TAB[5] (increment size for INC/handwheel)
  or
  SD41010 $SN_JOG_VAR_INCR_SIZE (size of the variable increment for JOG)
- Weighting of an increment:

  MD31090 $MA_JOG_INCR_WEIGHT
- Number of handwheel pulses per detent position:

  MD11320 $MN_HANDWH_IMP_PER_LATCH

For example, the axis traverses by 0.001 mm per handwheel detent position if machine function INC1 and the default setting of the above machine data are selected.

In the case of velocity override, the velocity results from the traverse path covered using the handwheel within a certain period of time.

Example

Assumptions:
The operator turns the handwheel with 100 pulses/second.

The selected machine function is INC100.

The default setting is made for the above machine data for handwheel weighting.

⇒ Handwheel traverse path per second:	10 mm
⇒ Velocity override:	0.6 m/min

PLC interface signals

As soon as the handwheel override takes effect, the following interface signals to the PLC are set to signal 1:

- For positioning axes / concurrent positioning axes / command axes / reciprocating axes:

  DB31, ... DBX62.1 (handwheel override active)
- For path axes:

  DB21, ... DBX33.3 (handwheel override active)

For the path input, depending on the traversing direction, the appropriate interface signals are output to the PLC:

- Machine axes:
  - DB31, ... DBX64.6/7 (traversing command minus/plus)
- Geometry axes:
  - DB21, … DBX40.6/7 (geometry axis 1: traversing command minus/plus)
  - DB21, … DBX46.6/7 (geometry axis 2: traversing command minus/plus)
  - DB21, … DBX52.6/7 (geometry axis 3: traversing command minus/plus)

Limitations

The axial limitations (software limit switch, hardware limit switch, working-area limitation) are effective in conjunction with handwheel override. With path default, the axis can be traversed with the handwheel in the programmed traversing direction only as far as the programmed target position.

The resulting velocity is limited by the axial machine data:

MD32000 $MA_MAX_AX_VELO(maximum axis velocity)

NC Stop/override = 0

If the feedrate override is set to 0% or an NC Stop is initiated while the handwheel override is active, the following applies:

- **For path definition:**

  The handwheel pulses arriving in the meantime are summated and stored. If NC Start or the feedrate override > 0%, the saved handwheel pulses become effective (i.e. are traversed).

  However, if the handwheel is first deactivated [via IS DB21, ... DBX12/16/20.0-2 (geometry axes 1/2/3: Activate handwheel (1, 2, 3))] then the stored handwheel pulses are deleted.

- **For velocity specification:**

  The handwheel pulses arriving in the meantime are not summated and are not active.

3.5.2 Programming and activating handwheel override

General information

When the handwheel override is programmed with NC language elements `FD` (for path axes) and `FDA` (for positioning axes), the following points must be observed:

- `FDA` and `FD` function **non-modally.**

  Exception for positioning axes: If traverse instruction `POSA` is programmed, the handwheel override can also act modally because this positioning axis does not affect the block transition.

- When the handwheel override is activated with `FDA` or `FD`, a **target position** must be programmed in the NC block for the positioning axis or for a path axis. When the programmed target position is reached, the handwheel override becomes inactive again.
- It is not possible to program `FDA` and `FD` or `FA` and `F` in the same NC block.
- The positioning axis must not be an indexing axis.

Positioning axis

Syntax for handwheel override: `FDA[AXi] = [feedrate value]`

Example 1:

Activate velocity override

```
N10 POS[U]=10 FDA[U]=100 POSA[V]=20 FDA[V]=150 . . .
```

`POS[U]=10`	Target position of positioning axis U
`FDA[U]=100`	Activate velocity override for positioning axis U; axis velocity of U = 100 mm/min
`POSA[V]=20`	Target position of positioning axis V (modally)
`FDA[V]=150`	Activate velocity override for positioning axis V; axis velocity of V = 150 mm/min

Example 2:

Activate path default and velocity override in the same NC block

```
N20 POS[U]=100 FDA[U]= 0 POS[V]=200 FDA[V]=150 . . .
```

`POS[U]=100`	Target position of positioning axis U
`FDA[U]= 0`	Activate path default for positioning axis U;
`POS [V]=200`	Target position of positioning axis V
`FDA[V]=150`	Activate velocity override for positioning axis V; axis velocity of V = 150 mm/min

Path axis

Syntax for handwheel override: `FD = [feedrate value]`

In order to activate "Handwheel override in automatic mode", the following requirements must have been met:

- Active movement commands from group 1: `G01`, `G02`, `G03`, `CIP`
- Exact stop active (`G60`)
- Linear feedrate in mm/min or inch/min active (`G94`)

These requirements are checked by the control and an alarm is output if any of them is not met.

Example 3:

Activate velocity override

```
N10 G01 X10 Y100 Z200 FD=1500 . . .
```

`X10 Y100 Z200`	Target position of path axes X, Y and Z
`FD=1500`	Activate velocity override for path axes; path velocity = 1500 mm/min

Concurrent positioning axis

The handwheel override for concurrent positioning axes is activated from the PLC via FC18 by setting the appropriate interface signal:

DB31, ... DBX62.1 (handwheel override active)

If the velocity parameter (F_Wert) is transferred with the value 0, then the activated handwheel override acts as distance input, i.e. in this case, the feed is not derived from the axial machine data (see also Section "P2: Positioning axes (Page 599)"):

MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)

References:

Function Manual, Basic Functions, Basic PLC Program (P3)

3.5.3 Special features of handwheel override in automatic mode

Velocity display

The velocity display for handwheel override shows the following values:

- Set velocity
  = programmed velocity
- Actual velocity
  = resultant velocity including handwheel override

Effect on transverse axes

If the axis is defined as a transverse axis and `DIAMON` is active, the handwheel pulses are interpreted and traversed as diameter values while handwheel override is active.

Dry-run feedrate

With active dry run
DB21, ... DBX0.6 (activate dry-run feedrate) = 1,
the dry-run feedrate is always effective
SD42100 $SC_DRY_RUN_FEED.

In this way, the axis is traversed to the programmed target position at dry-run feedrate without any influence from the handwheel despite the active handwheel override with path default (`FDA[AXi] = 0`), i.e., the path default is ineffective.

DRF active

When "Handwheel override in automatic mode" is activated, it is important to check whether the "DRF" function is active (DB21, ... DBX0.3 = 1).

If this were the case, the handwheel pulses would also cause a DRF offset of the axis. The user must, therefore, first deactivate DRF.

Feedrate override

The feedrate override does not affect the velocity of the movements produced by the handwheel (exception: 0%). It only affects the programmed feedrate.

With path default and fast handwheel movements, the axis may not be able to follow the handwheel rotation synchronously (especially in the case of a large handwheel-pulse weighting), causing the axis to overtravel.

3.6 Contour handwheel/path input using handwheel (option)

Function

When the function is activated, the feedrate of path and synchronized axes can be controlled via a handwheel in AUTOMATIC and MDI modes.

Availability

For the SINUMERIK 840D sl and SINUMERIK 828D systems, the "contour handwheel" function is available as an option that is under license.

Input mode (path or velocity input)

Either the distance or the velocity can be entered via the handwheel:

- **Path definition**

  Limiting the velocity to the maximum permissible value causes the axes to overtravel. The path defined by the handwheel is traversed and **no pulses are lost**.

- **Velocity specification**

  The handwheel only defines the traverse velocity. As soon as the handwheel stops, the axes stop too. Motion is braked immediately if no pulses are supplied from the handwheel in one IPO cycle, thus **preventing overtravel by the axes**. The handwheel pulses do not supply a path default.

The input mode is set with machine data:

MD11346 $MN_HANDWH_TRUE_DISTANCE (handwheel distance or velocity input)

Feedrate

The feedrate in mm/min is **dependent** on:

- The number of pulses supplied by the selected handwheel within one period
- Pulse evaluation of the handwheel via the machine data:

  MD11322 $MN_CONTOURHANDWH_IMP_PER_LATCH (contour handwheel pulses per detent position)

- The activated increment (INC1, 10, 100, etc.)
- The distance weighting of an increment of the first available geometry axis:

  MD31090 $MA_JOG_INCR_WEIGHT (evaluation of an increment for INC/handwheel)

The feedrate is **not dependent** on:

- The programmed feedrate mode (mm/min, mm/rev.)
- The programmed feedrate (resultant velocity can be higher)
- The rapid traverse velocity for G0 blocks
- The override (position 0% is effective, i.e. zero speed)

Traversing direction

The traversing direction depends on the direction of rotation:

- Clockwise

  → Results in travel in the programmed direction

  If the block-change criterion (IPO end) is reached, the program advances to the next block (response identical to `G60`).

- Counterclockwise

  → Results in travel in the programmed direction

  Here, the axes can only traverse to the appropriate block start. Pulses are not collected if the handwheel continues to rotate.

Activation of the function

The function can be activated via interface signals or via the NC program:

- Activation via interface signal

  Switching-in/switching-out is realized via the interface signal:

  DB21, ... DBX30.0-2 (activate contour handwheel (1, 2, 3))

- Activation via the NC program

  The contour handwheel can be activated in the NC program non-modally using `FD=0`, that is, velocity `F...` from the block before the contour handwheel applies in the following block **without** the need for additional programming.

  Note

  If no feedrate was programmed in the previous blocks, a corresponding alarm is output.

  `FD` and `F` cannot appear in the same NC block (triggers an alarm).

Contour-handwheel simulation

When the contour handwheel is activated, it can also be simulated.

After activation via interface signal
DB21, ... DBX30.3 (contour-handwheel simulation),
the feedrate is no longer defined by the contour handwheel; the programmed feedrate is used instead.

The direction is also defined via an interface signal:

DB21, ... DBX30.4 (negative direction simulation contour handwheel)

When the simulation is deselected or the direction is changed, the current movement is decelerated using a braking ramp.

Note

The override is effective as for NC-program execution.

Boundary conditions

- **Preconditions**
  Fixed feedrate, dry-run feedrate, thread cutting, or tapping must not be selected.
- **Limit values**
  The acceleration and velocity of the axes are limited to the values defined in the machine data.
- **Interruption of traversing movement**
  On NC Stop, the function remains selected but the handwheel pulses are not summated and are ineffective.
  Precondition: MD32084 $MA_HANDWH_CHAN_STOP_COND bit 2 = 1
  DRF
  A selected DRF function also has a path-override action.
- **Channel-specific deletion distance-to-go**
  This causes the movement triggered by the contour handwheel to be aborted; the axes are decelerated and the program is restarted with the next NC block. The contour handwheel then becomes effective again.

3.7 DRF offset

Function

The "DRF offset" function (differential resolver function) can be used to set an additive incremental zero offset in respect of geometry and auxiliary axes in the basic coordinate system in AUTOMATIC mode via an electronic handwheel.

The handwheel assignment, i.e. the assignment of the handwheel from which the increments for the DRF offset are to be derived, to the geometry or auxiliary axes that are to be moved by this, must be performed via the appropriate machine axes. The appropriate machine axes are those to which the geometry or auxiliary axis is mapped.

The DRF offset is not displayed in the axis actual-value display.

Applications

The DRF offset can be used, for example, in the following application cases:

- Offsetting tool wear within an NC block

  Where NC blocks have very long processing times, it becomes necessary to offset tool wear manually within the NC block (e.g. large surface-milling machines).
- Highly precise offset during grinding
- Simple temperature compensation

Note

The zero offset introduced via the DRF offset is always effective in all modes and after a RESET. It can, however, be suppressed non-modally in the part program.

Velocity reduction

The velocity generated using the handwheel for DRF can be reduced with respect to the JOG velocity:

MD32090 $MA_HANDWH_VELO_OVERLAY_FACTOR (ratio of JOG velocity to handwheel velocity (DRF))

DRF active

DRF must be active to allow the DRF offset to be modified by means of traversal with the handwheel. The following requirements must be fulfilled:

- AUTOMATIC mode
- DB21, ... DBX0.3 (activate DRF) = 1

The DRF offset can be activated/deactivated for specific channels using the "program control" function on the HMI user interface.

The HMI software then sets interface signal:
DB21, ... DBX24.3 (DRF selected) =1.

The PLC program (basic PLC program or user program) transfers this interface signal to interface signal
DB21, ... DBX0.3 (activate DRF) once the corresponding logic operation has been performed.

Control of DRF offset

The DRF offset can be modified, deleted or read:

User:	• Traversing with the handwheel
Part program:	• Reading via axis-specific system variable $AC_DRF[<axis>] • Deleting via parts-program command (`DRFOF`) for all axes in a channel • Non-modal suppression via parts-program command (`SUPA`) **References:** Programming Manual, Fundamentals
PLC user program:	• Reading the DRF offset (axis-specific) **References:** Function Manual, Basic Machine; PLC Basic Program (P3)
HMI user interface:	• Display of the DRF offset (axis-specific)

Note

If DRF offset is deleted, the axis is not traversed!

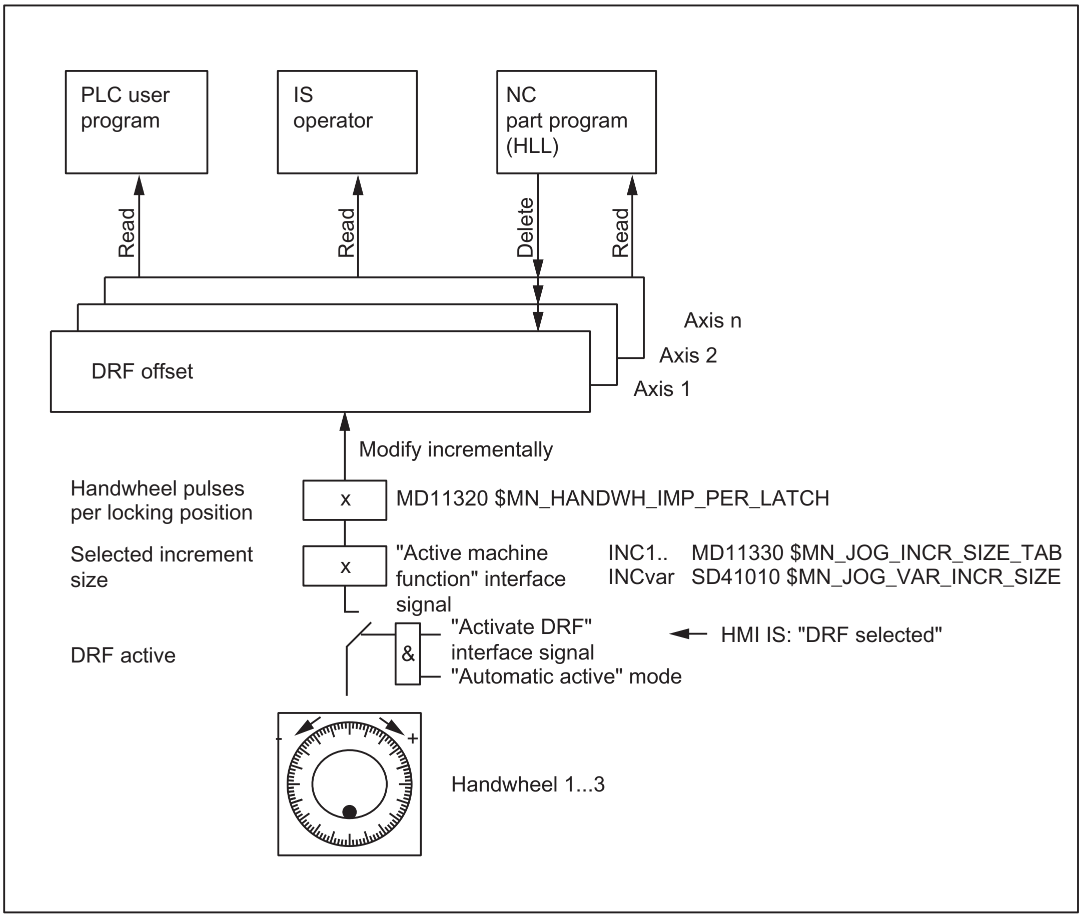


Figure 3-5 Control of DRF offset

Display

The axis actual-position display (ACTUAL POSITION) does not change while an axis is being traversed with the handwheel via DRF. The current axis DRF offset can be displayed in the DRF window.

Reference point approach

In phase 1 of the machine-axis reference point approach, the DRF offset for the corresponding geometry or auxiliary axis is deleted.

During the machine-axis reference point approach, a DRF offset for the corresponding geometry or auxiliary axis cannot be performed simultaneously.

Reset response

`PowerOn-Reset`: The DRF offset is deleted.

3.8 Approaching a fixed point in JOG

3.8.1 Introduction

Function

The machine user can use the "Approaching fixed point in JOG" function to approach axis positions defined through machine data by actuating the traverse keys of the machine control table. The traveling axis comes to a standstill automatically on reaching the defined fixed point.

Applications

Typical applications are, for example:

- Approaching a basic position before starting an NC program.
- Travel towards tool change points, loading points and pallet change points.

Requirements

- The "Approaching fixed point in JOG" can be activated only in the JOG mode.

  The function cannot be enabled in the JOG-REPOS and JOG-REF sub-modes and in JOG in the AUTOMATIC mode.
- The axis to be traversed must be referenced.
- A kinematic transformation may not be active.
- The axis to be traversed may not be a following axis of an active coupling.

Approaching a fixed point with G75

The process for approaching defined fixed points can also be activated from the part program using the `G75` command.

For more information on fixed point approach with `G75` please refer to:
References:
Programming Manual, Fundamentals; Section: "Additional commands" > "Approach fixed point (G75)".

3.8.2 Functionality

Procedure

Procedure in "Approaching fixed point in JOG"

- Selection of JOG mode
- Enabling the "Approach fixed point in JOG" function
- Traversing of the machine axis with traverse keys or handwheel

Activation

After selecting the "Approach fixed point in JOG" function, the PLC outputs the number of the fixed point to be approached binary coded to the NC using the following bits:

DB31, ... DBX13.0-2 (JOG approach fixed point)

The NC confirms the activation with the interface signal as soon as the function is effective:

DB31, ... DBX75.0-2 (JOG approach fixed point active)

Note

Activation is not possible:

- during an NCK reset
- In case of impending emergency stop
- During processing of an ASUP

No alarm message occurs. Delayed activation takes place after closure or after acknowledgement of the active function.

Sequence

The actual traversing is started with the traverse keys or the handwheel in the direction of the approaching fixed point.

The selected machine axis traverses till it comes to an automatic standstill at the fixed point.

The corresponding NC/PLC interface signal is reported on reaching the fixed point with "Exact stop fine":

DB31, ... DBX75.3-5 (JOG approach fixed point reached)

This display signal is also signaled if the axis reaches the fixed point position in the machine coordinates system via other methods e.g. NC program, FC18 (for 840D sl) or synchronized action on the setpoint side and comes to a standstill on the actual value side within the "Exact stop fine" tolerance window (MD36010 $MA_STOP_LIMIT_FINE).

Movement in the opposite direction

The response while traversing in the opposite direction, i.e.,against the direction of the approaching fixed point depends on the setting of Bit 2 in the machine data:

MD10735 $MN_JOG_MODE_MASK (settings for the JOG mode)

Traverse in the opposite direction is possible only if the bit is set.

Traverse in the opposite direction is blocked if the bit is not set and the following channel status message is output if an attempt is made with the traverse keys or with the handwheel to traverse in the direction opposite the approaching fixed point:

"JOG: <Axis> direction blocked"

Approaching other fixed point

The axis motion is cancelled and the following alarm is output if a different fixed point is selected while traversing to the fixed point:

Alarm 17812 "Channel %1 Axis %2 fixed point approach in JOG: Fixed point changed"

The message signal DB31, ... DBX75.0-2 (JOG - Approaching fixed point active) displays the number of the newly selected fixed point. The JOG traverse must be triggered again to continue traversing.

Note

To avoid the alarm message, the machine user should proceed as follows:

1. Cancel the current traverse movement with residual distance deletion.
2. Activate fixed point approach for another fixed point and start the operation after the axis comes to a standstill.

Withdrawal from fixed point / deactivation

To withdraw from a fixed position, you must deactivate the "Approaching fixed point in JOG" function. This is done by resetting the activation signal to "0".

DB31, ... DBX13.0-2 = 0

The message signals DB31, ... DBX75.0-2 (JOG - Approaching fixed point active) and DB31, ... DBX75.3-5 (JOG - Approaching fixed point reached) are deleted on leaving the fixed point position.

Special case: Axis is already on fixed point

The axis cannot be moved if, while starting the fixed point traverse, the axis is already at the position of the fixed point to be approached. This is displayed through the following channel status message:

"JOG: <Axis> position reached"

To withdraw from the fixed position, you must deactivate the "Approaching fixed point in JOG" function.

Special features of incremental travel

If, during incremental travel, the fixed point is reached before the increment is completed, then the increment is considered to have been completed fully. This is the case even when only whole increments are traveled.

MD11346 $MN_HANDWH_TRUE_DISTANCE = 2 or 3

Features of modulo rotary axes

Modulo rotary axes can approach the fixed point in both directions. The shortest path (DC) is not observed during the travel.

Features of spindles

A spindle changes to the positioning mode on actuating the "Approaching fixed point in JOG" function. The closed loop position control is active and the axis can traverse to the fixed point.

If no zero mark is detected the alarm message in the axis operation is output:

Alarm 17810 "Channel %1 Axis %2 not referenced"

As a spindle must also be a modulo rotary axis at all times, the same conditions apply for direction observation as for modulo rotary axes (refer to the paragraph "Features of modulo rotary axes")

3.8.3 Parameterization

Movement in the opposite direction

The behavior while traversing in the opposite direction - i.e. in the direction opposite to approaching the fixed point - depends on the following setting:

MD10735 $MN_JOG_MODE_MASK, bit 2 (settings for the JOG mode)

Bit	Value	Meaning
2	0	Travel in the opposite direction is **not** possible (default setting).
	1	Movement in the opposite direction is possible.

Fixed point positions

Up to four fixed point positions can be defined for an axis:

MD30600 $MA_FIX_POINT_POS[0...3] = <fixed point position 1...4>

Number of valid fixed point positions

The number of fixed point positions entered in MD30600 $MA_FIX_POINT_POS that are actually valid, can be defined using:

MD30610 $MA_NUM_FIX_POINT_POS = <number of valid fixed point positions>

Note

Exception: G75

For compatibility reasons, the following parameter assignment is also possible for G75:

MD30610 $MA_NUM_FIX_POINT_POS = 0 (no valid fixed point positions)

it is assumed that there are two valid fixed point positions in MD30600 $MA_FIX_POINT_POS[0] and [1].

Fixed point positions 1 and 2 can be activated via the NC/PLC interface, however they can only be approached via G75.

Axis dynamics

The axial acceleration and the axial jerk for "Approaching fixed point in JOG" are determined by the following machine data:

- When traversing with **traverse keys or handwheel**:
  - MD32301 $MA_JOG_MAX_ACCEL (maximum axial acceleration for JOG movements)
  - MD32436 $MA_JOG_MAX_JERK (maximum axial jerk for JOG movements)

  Note

  MD32436 $MA_JOG_MAX_JERK is only effective when the axial jerk limitation for single-axis movements has been enabled for the machine axes to be traversed:

  MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE

- During traversing via the **G75 part program command**:
  - MD32300 $MA_MAX_AX_ACCEL [0/1] (maximum axial acceleration for path motions in the dynamic response mode DYNNORM/DYNPOS)

  Note

  The type of positioning axis dynamics (DYNNORM or DYNPOS) is defined by the machine data:

  MD18960 $MN_POS_DYN_MODE = <mode>

– MD32431 $MA_MAX_AX_JERK [0] (maximum axial jerk for path motions in the dynamic response mode DYNNORM)

Note

MD32431 $MA_MAX_AX_JERK is only effective when the axial jerk limitation for single-axis movements has been enabled for the machine axes to be traversed:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE

Reference:
Function Manual, Basic Functions; Acceleration (B2)

3.8.4 Programming

System variables

The following system variables that can be read in the part program and in the synchronous actions for the "Approach fixed point" function.

System variable	Description
$AA_FIX_POINT_SELECTED [<Axis>]	Number of fixed point to be approached
$AA_FIX_POINT_ACT [<Axis>]	Number of the fixed point on which the axis is currently located

3.8.5 Supplementary Conditions

Axis is indexing axis

The axis is not traversed and an alarm is output if the axis to be traversed is an indexing axis and the fixed point position to be approached does not match an indexing position.

Frames active

All active frames are ignored. Traversing is performed in the machine coordinate system.

Offset values active

Active offset values (DRF, external zero offset, synchronized action offset $AA_OFF, online tool offset) are also traversed. The fixed point is a position in the machine coordinates system.

An alarm is issued if an offset movement (DRF, external zero offset, synchronized action offset $AA_OFF, online tool offset) is made during a fixed point approach in JOG. The position of the fixed point to be approached in the machine coordinates system is not reached; instead a position that would have been reached without active offset movement is reached. The NC/PLC interface signal DB31, ... DBX75.3-5 corresponding to the fixed point is not output.

Working-area limitations

Working-area limitations (in BCS and WCS) are considered and the axis motion is stopped on reaching the limits.

3.8.6 Application example

Target

A rotary axis (machine axis 4 [AX4]) is to be moved to Fixed Point 2 (90 degrees) with the "Approaching fixed point in JOG" function.

Parameter setting

The machine data for the "Approaching fixed point" function of machine axis 4 are parameterized as follows:

MD30610 $MA_NUM_FIX_POINT_POS[AX4] = 4	4 fixed points are defined for machine axis 4.
MD30600 $MA_FIX_POINT_POS[0,AX4] = 0	1st Fixed point of AX4 = 0 degree
MD30600 $MA_FIX_POINT_POS[1,AX4] = 90	2nd Fixed point of AX4 = 90 degree
MD30600 $MA_FIX_POINT_POS[2,AX4] = 180	3rd Fixed point of AX4 = 180 degree
MD30600 $MA_FIX_POINT_POS[3,AX4] = 270	4th Fixed point of AX4 = 270 degree

Initial situation

Machine axis 4 is referred and is in Position 0 degree. This corresponds to the 1st fixed position and is output through the NC/PLC interface signal:

DB31 DBX75.0 = 1 (Bit 0-2 = 1)

Approaching fixed point 2

The control is switched in the JOG mode.

The "Approaching fixed point" function is activated on Fixed Point 2 via the NC/PLC interface signal:

DB31 DBX13.1 = 1 (Bit 0-2 = 2)

The actuation is confirmed via the NC/PLC interface signal:

DB31 DB75.1 = 1 (Bit 0-2 = 2)

The Plus traverse key in the machine control table is used to traverse continuously to approach Fixed Point 2.

The machine axis 4 stops at the 90 degree position. This is reported via the NC/PLC interface signal:

DB31 DBX75.4 = 1 (Bit 3-5 = 2)

3.9 Retraction in the tool direction (JOG retract)

3.9.1 Overview

Function

The function "Retraction in the tool direction in the JOG retract submode", called "JOG retract" in the following, supports the manual tool retraction in the workpiece coordinate system (WCS) after a program abort through a power off of the control or a channel reset in the AUTOMATIC or MDI mode.

In particular, the specific features of the following functions are taken into account:

- Tapping with compensating chuck and speed-controlled spindle with encoder (`G33`)
- Tapping without compensating chuck and position-controlled spindle (`G331`, `G332`)
- Machining with tool orientation with swivel cycle CYCLE800 or orientation transformation

Availability

The JOG retract function can be selected only when:

- Tapping using G33/G331/G332

or general:

- A machining with a drilling tool (tool type 200 to 299)

has been interrupted with a channel reset or power off.

Data to be restored

In order to be able to execute the retraction in the tool direction after a program abort, the following data which was active in the channel before the program abort, is restored:

- Active tool offset
- Active machining plane
- Active tool carrier
- Active transformation data block with transformation parameters
- Data of the thread group with G33 or G331/G332
- Positions of the axes that are involved in the transformation

If this data is completely available after a program abort, it is restored in the channel when the JOG retract submode is selected. In this case, the workpiece coordinate system (WCS) is aligned by the control in such a way that one of the geometry axes is in the direction of the tool axis. The tool retraction can then be performed manually by traversing this geometry axis.

Note

Data backup

The specified data itself is not backed up, only the references to this data. If the data is changed prior to the selection of JOG retract, the function is performed on the basis of the changed data.

Restrictions

Programs that can be started by interrupt signals are **not** executed in JOG retract.

3.9.2 Parameter assignment

NC-specific machine data

Automatic selection of JOG retract after Power On

After the control has run up (Power On), the channels are in the parameterized default mode:

MD10720 $MN_OPERATING_MODE_DEFAULT[<mode group>] = <default mode>

The following machine data can be set so that when the control has run up, JOG mode is selected irrespective of the parameterized default mode. However, the selection is only performed when the mode group retraction data is available in a channel.

MD10721 $MN_OPERATING_MODE_EXTENDED[<mode group>] = <value>	
Value	Meaning
0	Selection of the operating mode corresponding to MD10720 $MN_OPERATING_MODE_DEFAULT
1	Selection of JOG mode, if: DB21, ... DBX377.5 == 1 ("retraction data available")

Enable of the traversing direction

The retraction can be limited to the positive direction of travel or enabled for both travel directions:

MD10735 $MN_JOG_MODE_MASK, Bit 8	
Value	Meaning
0	Traversing enable: Only plus direction
1	Traversing enable: Plus and minus direction

Axial machine data

Measuring system status

The current status of the measuring system is displayed via the machine data, or must be set in preparation for the required behavior:

MD34210 $MA_ENC_REFP_STATE[<axis>]	
Value 1)	Meaning
Absolute encoder	
2	Encoder is calibrated
Incremental encoder	
1	Automatic referencing: Enabled, but encoder still not referenced
2	Automatic referencing: Encoder is referenced and in exact stop, automatic referencing takes effect at next time encoder activation
3	Restoration of the actual position: The last axis position buffered before switching off is restored, no automatic referencing
1) Only values relevant for JOG retract	

3.9.3 Selection

Function

Requirement

The selection of JOG retract is only possible, if valid retraction data is available for the relevant channel, the channel is in JOG mode and in the "Reset" state:

- DB21, ... DBX377.5 == 1 (retraction data available)
- DB11, ... DBX(n*20+6).2 == 1 (active mode: JOG, with n=0, 1 ,2 ... for mode group 1, 2, ...)
- DB21, … DBX35.7 == 1 (channel state: Reset).

Axes and spindles

All active measuring systems of the machine axes involved in the retraction in the tool direction must be in the "referenced" or "restored" state when the control is switched on again (Power On).

A detailed description of the automatic restoration of actual positions after the next start of the control (Power On) can be found in:

References:
Basic Functions Function Manual, Section "R1 Referencing" > "Automatic restoration of the machine reference"

Axis interchange

If all the axes and spindles involved in the retraction are not assigned to the channel at the time of selection, an implicit axis interchange is performed for the missing axes.

Coordinate system

With the selection of JOG retract, the workpiece coordinate system (WCS) is set for the traversing of the channel axes. Axes involved in the retraction can only be traversed in the WCS. All axes that are not involved in the retraction can also be traversed in the machine coordinate system (MCS).

Selection options

The selection of JOG retract can be supported through automatic selection of the JOG mode after Power On. The actual selection of JOG retract is then performed through manual selection via the user interface or via the PLC user program.

Automatic selection of JOG mode

If retraction data is available for a channel, JOG mode is selected for the mode group after the control runs up. The parameter assignment is performed via machine data MD10721 $MN_OPERATING_MODE_EXTENDED. See Section "Parameter assignment (Page 191)".

Selection via the user interface

The selection of JOG retract is performed on the user interface via the "Retract" softkey:

"Machine operating area" > "ETC key (">")" > "Retract"

Selection by PLC user program

The following actions must be performed to select JOG retract by the PLC user program:

- Channel-specific query whether retraction data is available
  DB21, … DBX377.5 == 1 (retraction data available)
- Mode group-specific selection of JOG mode:
  DB11, ... DBX(n*20+6).2 == 1 (active mode: JOG, with n=0, 1 ,2 ... for mode group 1, 2, ...)
- Channel-specific selection of JOG retract via the PI service "RETRAC".
  A detailed description of the activation of the PI service "RETRAC" via the function block FB 4 can be found in:
  References:
  Basic functions Function Manual, Section "P3: Basic PLC program for SINUMERIK 840D sl" > "PI services"
- After confirmation of the selection of JOG retract (see below), select the workpiece coordinate system:
  - DB19.DBX20.7 (switch MCS/WCS)
  - DB19.DBX0.7 (actual value 1 =WCS, 0 =MCS)

Selection confirmation

Following a successful selection, the following NC/PLC interface signal is set:

DB21, ... DBX377.4 = 1 (JOG retract active)

Thread cutting (`G33` or `G331`/`G332`)

If the program execution was aborted during a thread cutting operation (`G33` or `G331`/`G332`), the axis grouping of tool axis and spindle is restored when JOG retract is selected. The controller parameters and parameter sets for the involved axes are also set in accordance with a programmed thread cutting in the part program.

3.9.4 Tool retraction

General retraction behavior

The tool is retracted by manually traversing the geometry axis specified when selecting JOG retract in the workpiece coordinate system (WCS). The specification of the retraction movement can be performed via the traversing keys of the machine control panel (MCP) or via the handwheel. Retraction is possible within the traversing range limits (working area limitation, software limit switch, etc.).

The retraction movement can be stopped and started again with NC stop and NC start.

The axes and spindles not involved in the tool retraction can be manually traversed as required.

Traversing direction

Per default, the retraction movement is only enabled for the positive traversing direction. If traversing in the negative direction is also to be possible, this must be explicitly enabled:

MD10735 $MN_JOG_MODE_MASK, Bit 8

Retraction behavior for thread cutting (G33) or tapping (G331, G332)

In addition to the geometry axes, a spindle is also involved in the retraction movement for thread cutting (`G33` or `G331`/`G332`). The retraction movement is executed when either the selected geometry axis or the spindle is traversed.

Retraction using the handwheel

If one of the axes involved in the retraction movement is traversed by means of traversing key, the handwheel pulses for other axes involved in the retraction movement are ignored.

If handwheels have been selected for several axes and these are moved, the handwheel pulses are evaluated in the following order:

1. Retraction axis
2. Spindle
3. Axes/spindles not involved in the retraction movement

The pulses of the other handwheels are ignored and only evaluated at standstill of the preceding handwheels.

Locked functions

During JOG retract, the requirements for the following functions are ignored:

- Spindle start via DB31, ... DBX30
- Traversing of a spindle or axis involved in the retraction via function block FC18
- Switchover of a spindle or an axis involved in the retraction to the PLC-controlled axis
- Changing a spindle or axis involved in the retraction to another channel
- Using a spindle or axis involved in the retraction as main run axis (command axis, oscillating axis, FC18 / concurrent axis)

3.9.5 Deselection

JOG retract is deselected channel-specifically via:

- Channel reset
  - Machine control panel: "Reset" button
  - Basic PLC program: DB21, … DBX7.7 = 1 (reset)
- User interface: "Back" ("<<") softkey

This activates the initial settings via the following machine data:

- MD20110 $MC_RESET_MODE_MASK[<channel>] ()
- MD20150 $MC_GCODE_RESET_VALUES[<channel>] ()
- MD20151 $MC_GCODE_RESET_MODE[<channel>] ()

After the deselection, the channel is in the JOG mode in the channel state "Reset".

All channel axes can now be traversed manually in JOG mode. The retraction data is retained. It is therefore possible to select JOG retract again.

NC/PLC interface signal

With the deselection of JOG retract, the NC/PLC interface signal is reset:

DB21, ... DBX377.4 = 0 (JOG retract active)

3.9.6 Repeated selection

JOG retract can continue to be selected as long as retraction data is available:

DB21, ... DBX377.5 == 1 (retraction data available)

If selected again, the original retraction data is restored. In each case, a different geometry axis can be specified as retraction axis.

Changed axis positions

The channel axes can be traversed in JOG mode while JOG retract is deselected. If JOG retract is selected again, the retraction movement is executed on the basis of the new axis positions.

3.9.7 Continuing machining

AUTOMATIC mode

Before the aborted part program is continued with NC start in AUTOMATIC mode, all machine axes with active measuring systems in the state "restored" or "not referenced" must be referenced.

NOTICE
Possible axis position displacement
After the control runs up after a power failure, the axis positions for incremental measuring systems are synchronized or restored corresponding to the setting in machine data. Once the tool has been retracted in JOG retract submode, axes whose positions have been restored are referenced.

MDI mode and overstore

In the MDI mode and for the overstore function, machining can also be performed, without referencing the axes, with restored positions. To do this, NC start with restored positions must be enabled explicitly for a specific channel:

MD20700 $MC_REFP_NC_START_LOCK = 2

Block search at interruption point

The processing of the part program can be continued from the interruption point via a block search at the interruption point. The last block of the main program level before the interruption is available as interruption point.

A detailed description of the function and for the operation of the block search can be found in:

References

Basic Functions Function Manual; Section "K1: Mode group, channel, program operation, reset response" > "Block search" or "Block search type 5 SERUPRO"

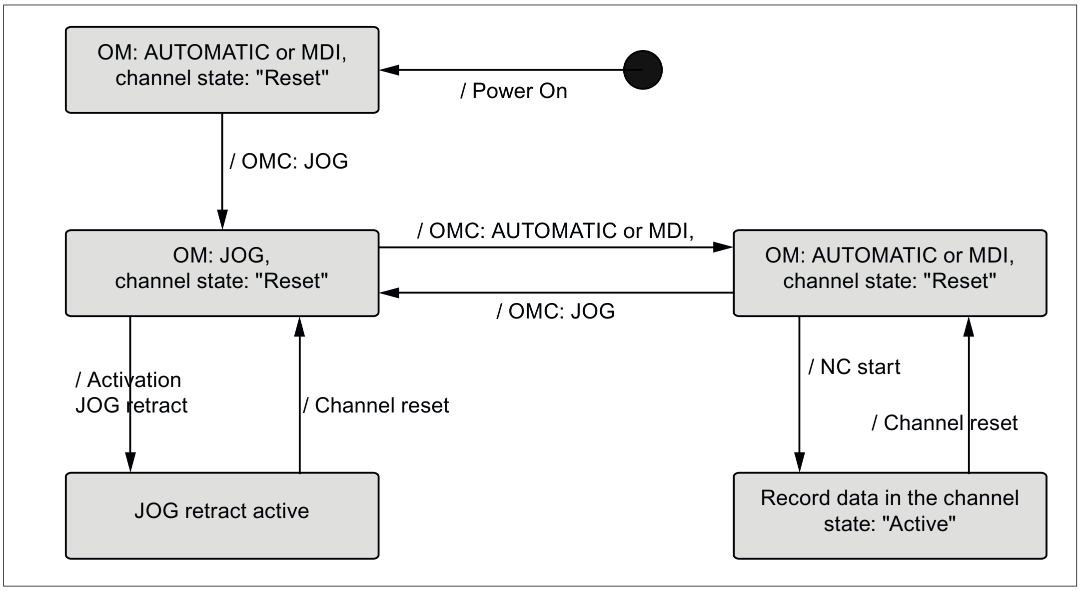
Turning or Milling Operating Manual; Section "Workpiece machining" > "Starting machining at a specific point"

Continue with NC start

With NC start in the AUTOMATIC or MDI mode, the processing of the part program is continued from the selected position. The following actions are performed with regard to JOG retract:

- The retraction data is deleted.
- The NC/PLC interface signal is reset:
  DB21, … DBX377.5 = 0 (retraction data available)
- The current data environment is saved.

3.9.8 State diagram



OM Operating mode
OMC Operating mode change

Figure 3-6 State diagram: JOG retract

3.9.9 System data

The following system data is available for JOG retract:

Meaning	System variable $VA_	NC/PLC interface	OPI variable
Retraction data available	-	DB21, ... DBX377.5	retractState, bit 0
JOG retract active	-	DB21, ... DBX377.4	retractState, bit 1
Retraction axis	-	-	retractState, bits 2 - 3
Position restored, 1st measuring system	$AA_POSRES	DB31, ... DBX71.4	aaPosRes
Position restored, 2nd measuring system	$AA_POSRES	DB31, ... DBX71.5	aaPosRes

References

OPI and system variable

Lists Book 2 List Manual, Section "Variables".

3.9.10 Supplementary conditions

Incremental measuring systems

The user must ensure that machine axes with incremental measuring systems are clamped with sufficient speed in the event of a power failure to prevent a change to the last position, known and saved by the control. Otherwise the assumed position when the control is restarted differs greatly to the actual position of the machine axis. Drive-autonomous retraction must also not be activated for these axes.

A detailed description of the automatic restoration of actual positions after the next start of the control (Power On) can be found in:

References:
Basic Functions Function Manual, Section "R1 Referencing" > "Automatic restoration of the machine reference"

Drive-autonomous retraction

The "Drive-autonomous retraction" function must not be activated for machine axes involved in the retraction movement.

Axis couplings

Axis couplings are not restored with the selection of JOG retract.

Tapping using G63

JOG retract is not possible for tapping with compensating chuck and speed-controlled spindle without encoder (`G63`).

Transformations

If a transformation is selected through JOG retract, the active measuring systems of all machine axes involved in the transformation must be in the "referenced" or "restored" state.

OEM transformations such as parallel kinematics for hexapods, can only be traversed with referenced measuring systems. Traversing with restored axis positions is not possible.

Tool orientation via directly programmed orientation axes

JOG retract cannot generate a retraction movement in the tool direction if the tool orientation is not performed via NC functions, but through direct programming of the orientation axes.

NCU link

JOG retract is also possible in conjunction with NCU-wide traversing of axes (see Section "NCU link (Page 79)"). The state of axis containers is not changed through JOG retract. Adaptations required for a retraction movement in the tool direction must be performed by the user before traversing in JOG retract, e.g. in the MDI mode.

3.10 Start-up: Handwheels

3.10.1 General information

In order to operate handwheels of a SINUMERIK control system, they have to be parameterized via NCK machine data.

If the handwheels are not directly connected to the control, additional measures are required, e.g. connection via PROFIBUS- or Ethernet-MCP or handwheel module, inserting and configuring the module with SIMATIC STEP 7, HW-Config.

Note

Currently only six handwheels can be parameterized in a SINUMERIK control system.

Connection options

SINUMERIK 840D sl

For SINUMERIK 840D sl, handwheels can be connected via the following components:

- PROFIBUS (Page 203) module
- Ethernet (Page 206) module

Note

Several handwheels, which are connected via different components, can be connected to one SINUMERIK 840D sl at the same time.

SINUMERIK 828D

For SINUMERIK 828D, handwheels can be connected via the following components:

- PPU (Page 201)
- Machine control panel (MCP) via PROFIBUS (Page 202)

Note

Several handwheels, which are connected via different components, can be connected to one SINUMERIK 828D at the same time.

3.10.2 Connection via PPU (only 828D)

Parameter assignment

Handwheels directly connected to terminal X143 of the PPU are parameterized using the following NCK machine data:

- MD11350 $MN_HANDWHEEL_SEGMENT[<*Handwheel_No._in_NCK - 1*>] = 2

  When directly connected to the PPU, a 2 (8xxD_HW) must always be entered as handwheel segment.

- MD11351 $MN_HANDWHEEL_MODULE[< *Handwheel_No._in_NCK - 1* >] = 1

  When directly connected to the PPU, a 1 must always be entered.

- MD11352 $MN_HANDWHEEL_INPUT[< *Handwheel_No_in_NCK - 1* >] = <*Handwheel connection* >

  The number of the handwheel has to be entered: 1 or 2

> **Note**
>
> Two handwheels can be connected to a PPU (terminal X143).

Example

Parameter assignment of two handwheels connected via terminal X143 of the PPU.

Machine data	Value	Meaning
		1st handwheel:
MD11350 $MN_HANDWHEEL_SEGMENT[0]	2	Hardware segment: 8xxD_HW
MD11351 $MN_HANDWHEEL_MODULE[0]	1	---
MD11352 $MN_HANDWHEEL_INPUT[0]	1	1st handwheel connection of X143
		2nd handwheel:
MD11350 $MN_HANDWHEEL_SEGMENT[1]	2	Hardware segment: 8xxD_HW
MD11351 $MN_HANDWHEEL_MODULE[1]	1	---
MD11352 $MN_HANDWHEEL_INPUT[1]	2	2nd handwheel connection of X143

3.10.3 Connection via PROFIBUS (828D)

Parameter assignment

For the SINUMERIK 828D, in addition to the connection of two handwheels to the PPU interface, terminal X143, a third handwheel can also be connected via a machine control panel, e.g. MCP 483C PN, interface X60.

The parameter assignment of the **third** handwheel is performed in the following NCK machine data:

- MD11350 $MN_HANDWHEEL_SEGMENT[2] = 5
- MD11351 $MN_HANDWHEEL_MODULE[2] = 1
- MD11352 $MN_HANDWHEEL_INPUT[2] = 1

Requirement

Operation of the control with default data (machine data, STEP 7 configuration).

Example

Parameterization of three handwheels, connected via PPU and an "MCP 483C PN" machine control panel.

Handwheel number in NCK	Machine data set (Index)	Connection
1	0	PPU, 1st handwheel in the handwheel slot
2	1	PPU, 2nd handwheel in the handwheel slot
3	2	MCP, 1st handwheel in the handwheel slot

Parameterizing in the NCK machine data:

Machine data	Value	Meaning
		1st handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[0]	2	Hardware segment: 8xxD_HW
MD11351 $MN_HANDWHEEL_MODULE[0]	1	---
MD11352 $MN_HANDWHEEL_INPUT[0]	1	1st handwheel in the handwheel slot
		2nd handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[1]	2	Hardware segment: 8xxD_HW
MD11351 $MN_HANDWHEEL_MODULE[1]	1	---
MD11352 $MN_HANDWHEEL_INPUT[1]	2	2nd handwheel in the handwheel slot
		3rd handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[2]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[2]	1	Reference to logical base address of the handwheel slot of the MCP
MD11352 $MN_HANDWHEEL_INPUT[2]	1	1st handwheel in the handwheel slot

3.10.4 Connection via PROFIBUS (840D sl)

Parameter assignment

The parameter assignment of handwheels connected via PROFIBUSmodules (e.g. "MCP 483" machine control table) is performed with the following NCK machine data:

- MD11350 $MN_HANDWHEEL_SEGMENT[<*Handwheel_No._in_NCK - 1*>] = 5

  When connected via PROFIBUSmodule, the hardware segment has always to be entered as 5 (PROFIBUS).
- MD11351 $MN_HANDWHEEL_MODULE[<*Handwheel_No._in_NCK - 1*>] = <*Index + 1*>

  The reference to the MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS[<*Index*>]

  has to be entered, which contains the logical base address of the handwheel.
- MD11352 $MN_HANDWHEEL_INPUT[<*Handwheel_No._in_NCK - 1*>] = <*Number_in_handwheel_slot*>

  A handwheel slot can contain several handwheels. The number of the handwheel within the handwheel slot has to entered: 1, 2, ...
- MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS[<*Index*>] = <*logical base address*>

  The logical base address of the handwheel slot, specified in SIMATIC STEP 7, HW Config, has to be entered.

Handwheel slot

The PROFIBUSmodule must be configured besides the parameterization of handwheels in the NCK machine data in STEP 7. Among others the logical address of the handwheel slot is specified.

The handwheel slot is situated at the following slot of the PROFIBUSmodule:

PROFIBUS module	Slot
Machine control panel MCP 483	2
Machine control panel MCP 310	2
Handwheel connection module	1

Example

Parameterization of five handwheels, connected via four machine control tables "MCP 483". Two handwheels can be connected to a machine control table "MCP 483".

Handwheel number in NCK	Machine data set (Index)	Connection
1	0	1st MCP, 1st handwheel in handwheel slot
2	1	1st MCP, 2nd handwheel in the handwheel slot

Handwheel number in NCK	Machine data set (Index)	Connection
3	2	2nd MCP, 1st handwheel in the handwheel slot
5	4	3rd MCP, 1st handwheel in the handwheel slot
6	5	4th MCP, 2nd handwheel in the handwheel slot

The 4th handwheel in the NCK is not used (gap in machine data).

Note

Machine data gaps are allowed when parameterizing handwheels in NCK machine data.

Machine control tables have been configured in SIMATIC STEP 7, HW Config as follows:

	Slot	DP ID	Order No. / Description ...	I address	O address
1st MCP	1	55	Standard+Handwheel	0 ... 7	0 ... 7
	2	*2AE*	*→ standard+handwheel*	*288 ... 291*	
	3	*1*	*→ standard+handwheel*		
2nd MCP	1	55	Standard+Handwheel	8 ... 15	8 ... 15
	2	*2AE*	*→ standard+handwheel*	*304 ... 307*	
	3	*1*	*→ standard+handwheel*		
3rd MCP	1	55	Standard+Handwheel	16 ... 23	16 ... 23
	2	*2AE*	*→ standard+handwheel*	*320 ... 323*	
	3	*1*	*→ standard+handwheel*		
4th MCP	1	55	Standard+Handwheel	24 ... 29	24 ... 29
	2	*2AE*	*→ standard+handwheel*	*330 ... 333*	
	3	*1*	*→ standard+handwheel*		

Parameterizing in the NCK machine data:

Machine data	Value	Meaning
		1st handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[0]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[0]	1	Reference to logical base address of the handwheel slot of the 1st MCP
MD11352 $MN_HANDWHEEL_INPUT[0]	1	1st handwheel in the handwheel slot
		2nd handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[1]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[1]	1	Reference to logical base address of the handwheel slot of the 1st MCP
MD11352 $MN_HANDWHEEL_INPUT[1]	2	2nd handwheel in the handwheel slot

Machine data	Value	Meaning
		3rd handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[2]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[2]	2	Reference to logical base address of the handwheel slot of the 2nd MCP
MD11352 $MN_HANDWHEEL_INPUT[2]	1	1st handwheel in the handwheel slot
		4th handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[3]	0	No handwheel parameterized
MD11351 $MN_HANDWHEEL_MODULE[3]	0	No handwheel parameterized
MD11352 $MN_HANDWHEEL_INPUT[3]	0	No handwheel parameterized
		5th handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[4]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[4]	6	Reference to logical base address of the handwheel slot of the 3rd MCP
MD11352 $MN_HANDWHEEL_INPUT[4]	1	1st handwheel in the handwheel slot
		6th handwheel in the NCK
MD11350 $MN_HANDWHEEL_SEGMENT[5]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[5]	5	Reference to logical base address of the handwheel slot of the 4th MCP
MD11352 $MN_HANDWHEEL_INPUT[5]	2	2nd handwheel in the handwheel slot

Logical base addresses:

Machine data	Value	Meaning
MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS [0]	288	Logical base address handwheel slot 1st MCP
MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS [1]	304	Logical base address handwheel slot 2nd MCP
MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS [4]	330	Logical base address handwheel slot 4th MCP
MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS [5]	320	Logical base address handwheel slot 3rd MCP

3.10.5 Connected via Ethernet (only 840D sl)

Parameter setting

The parameters for handwheels connected via Ethernet modules, e.g. machine control panel "MCP 483C IE", "HT 8", or "HT 2", are assigned in the following NC machine data:

- MD11350 $MN_HANDWHEEL_SEGMENT[< x - 1 >] = 7

  When connected via Ethernet modules, the segment always has to be entered as 7 (Ethernet).
- MD11351 $MN_HANDWHEEL_MODULE[< x - 1 >] = 1

  When connected via Ethernet modules, the module always has to be entered as 1.
- MD11352 $MN_HANDWHEEL_INPUT[< x - 1 >] = y

  where y = 1, 2, 3, etc. (handwheel interface at the Ethernet bus)

where x = 1, 2, 3, etc. (handwheel number in the NC)

Handwheel interfaces at the Ethernet Bus

The handwheel interfaces at the Ethernet bus are numbered on the basis of the following considerations:

- The sequence of the operator component interfaces is: MCP1, MCP2, BHG
- Each operator component interface has two handwheel interfaces
- Operator components: MCP 483C IE

  A maximum of two handwheels can be connected to an MCP 483C IE via connections X60 and X61 . The assignment of the connections in the operator component interface is:
  - Connection X60: 1st handwheel in operator component interface MCP1 /MCP2
  - Connection X61: 2nd handwheel in operator component interface MCP1 /MCP2
- Operator components: HT 8

  The handwheel of the HT 8 is always assigned to the 1st handwheel of operator component interface MCP1 /MCP2 .
- Operator components: HT 2

  The handwheel of the HT 2 is always assigned to the 1st handwheel of operator component interface BHG .

Operator component interface ->	MCP1		MCP2		BHG	
Handwheel interface 1)	1	2	1	2	1	2
FB1 parameters 2)	MCP1BusAdr		MCP2BusAdr		BHGRecGDNo	
Assignment of the handwheels 3)						
MCP 483C IE	X60	X61	X60	X61	-	-
HT 8	x	-	x	-	-	-
HT 2	-	-	-	-	x	-
Handwheel interface at the Ethernet bus (y) 4) ->	1	2	3	4	5	6

1) Numbering of the handwheel interfaces within an operator component interface
2) Assignment of the operator component to the interface via the corresponding FB1 parameter
3) Assignment of the handwheels of the respective operator components to the handwheel interfaces
4) Numbering of the handwheel interfaces at the Ethernet bus -> MD11352 $MN_HANDWHEEL_INPUT[< x - 1 >] = y

Example

Parameterization of 3 handwheels, connected via the following operator components:

Operator component interface ->	MCP1		MCP2		BHG	
Operator component	HT 8		MCP 483C		HT 2	
FB1 parameters	MCP1BusAdr := 39		MCP2BusAdr := 192		BHGRecGDNo := 40	
Handwheel interface	**x**	-	-	**X61**	**x**	-
Handwheel interface at Ethernet Bus ->	**1**	2	3	**4**	**5**	6

Table 3- 1 NCK machine data for the handwheel assignment

Machine data	Value	Description
		HT 8: Handwheel number in the NC = 1
MD11350 $MN_HANDWHEEL_SEGMENT[0]	7	Segment: Ethernet
MD11350 $MN_HANDWHEEL_MODULE[0]	1	Module: Ethernet
MD11350 $MN_HANDWHEEL_INPUT[0]	1	Handwheel interface at Ethernet bus
		MCP 483C IE: Handwheel number in the NC = 2
MD11350 $MN_HANDWHEEL_SEGMENT[1]	7	Segment: Ethernet
MD11350 $MN_HANDWHEEL_ MODULE [1]	1	Module: Ethernet
MD11350 $MN_HANDWHEEL_ INPUT [1]	4	Handwheel interface at Ethernet bus
		HT 2: Handwheel number in the NC = 3
MD11350 $MN_HANDWHEEL_SEGMENT[2]	7	Segment: Ethernet
MD11350 $MN_HANDWHEEL_ MODULE [2]	1	Module: Ethernet
MD11350 $MN_HANDWHEEL_ INPUT [2]	5	Handwheel interface at Ethernet bus

Table 3- 2 FB1 parameters (excerpt)

Parameter	Value	Remark
MCPNum	:= 2	// Number of connected MCP
		// MCP1 = HT 8
MCP1In	...	// MCP1-Parameter ...
...	...	
MCP1BusAdr	:= 39	// Via switches S1 and S2 on the connecting device
		// set "IP address"
		// MCP2 = MCP 483C IE
MCP2In	...	// MCP2-Parameter ...
...	...	
MCP2BusAdr	:= 192	// Via switch S2 on the MCP 483C
		// set "IP address"
MCPBusType	:= b#16#55	// Bus type: Ethernet
		// HHU = HT 2
HHU	:= 5	// Bus type: Ethernet
HHUIn	...	// HHU Parameter ...
...	...	
HHURecGDNo	:= 40	// Via switches S1 and S2 on the connecting device
		// set "IP address"

Filter time

Since the handwheel pulses on the Ethernet bus are not transferred deterministically, filtering (smoothing) of the handwheel pulse transfer process may be necessary for highly dynamic drives. The parameter for the filter time is assigned using the following machine data:

- MD11354 $MN_HANDWHEEL_FILTER_TIME[< x - 1 >] = <filter time>

  where x = 1, 2, 3, etc. (handwheel number in the NC) and filter time = 0.0 to 2.0 s

The filter time specifies the time it takes for the handwheel pulses transferred to the control to be sent on to the interpolator for traversing purposes. With a filter time of 0.0 s, the handwheel pulses are sent on to the interpolator within a single interpolation cycle. This can result in the relevant axis being traversed jerkily.

The recommended filter time is 0.2 to 0.5 s.

Stationary state detection

A stationary state is detected by the Ethernet modules to which the handwheel is connected. If a handwheel does not transfer any handwheel pulses for a defined period of time, the module detects this to be a stationary state and transfers it to the NC/PLC interface:

NC/PLC interface signal	Value	Description
DB10, DBX245.0	0	Handwheel 1 is operated
	1	Handwheel 1 is stationary
DB10, DBX245.1	0	Handwheel 2 is operated
	1	Handwheel 2 is stationary
DB10, DBX245.2	0	Handwheel 3 is operated
	1	Handwheel 3 is stationary

By evaluating the signal, it is possible to reduce the overtravel of an axis traversed via the handwheel, due to the handwheel pulses that have been collected in the control but not yet transferred to the interpolator for traversing purposes. To do this, deletion of the distance-to-go must be triggered for the relevant axis or in the channel when a stationary state is detected:

- DB31,... DBX2.2 = 1 (axial deletion of distance-to-go)
- DB21,... DBX6.2 = 1 (channel-spec. deletion of distance-to-go)

3.11 Special features relating to manual and handwheel travel

3.11.1 Manual travel of geometry/orientation axes

Coordinate systems in JOG

In JOG mode, the user can also traverse the axes declared as geometry axes in the workpiece coordinate system manually. Any coordinate offsets or rotations that have been selected remain active.

Note

In the JOG mode, using the "Handling transformation package" for SINUMERIK 840D sl, the translation of geometry axes in several valid references systems can be set separately from one another.

Reference:
Function Manual, Special Functions; Multi-Axis Transformations (F2), Section: "Cartesian manual travel"

Application

Manual movements for which transformations and frames have to be active. The geometry axes are traversed in the most recently valid coordinate system. The special features of geometry-axis manual travel are described below.

Simultaneous travel

Only one geometry axis can be traversed continuously or incrementally at one time using the traversing keys. Where an attempt is made to traverse more than one geometry axis, alarm 20062 "Axis already active" is output. However, three geometry axes can be traversed simultaneously using handwheels 1 to 3. Alarm 20060 is output if only one axis is not defined as a geometry axis.

PLC interface

For geometry/orientation axes, there is a separate PLC interface that contains the same signals as the axis-specific PLC interface:

- Geometry axes:

  DB21, ... DBB12-23 and

  DB21, ... DBB40-56
- Orientation axes:

  DB21, ... DBB320-331 and

  DB21, ... DBB332-343

Feedrate/rapid traverse override

The channel-specific feedrate-override switch and rapid-traverse-override switch are active for geometry-axis manual travel in rapid traverse override.

Acceleration and jerk

For the manual travel of geometry axes/orientation, the acceleration and jerk can be limited for specific channels. This enables better handling of the kinematics that generate Cartesian motions entirely via rotary axes (robots).

Geometry axes

The maximum acceleration when manually traversing geometry axes can be specified for each channel via the machine data:

MD21166 $MC_JOG_ACCEL_GEO [<geometry axis>]

With <geometry axis> = 0, 1, 2

The maximum jerk when manually traversing geometry axes in the SOFT acceleration mode (acceleration with jerk limitation) can be specified for each channel via the machine data:

MD21168 $MC_JOG_JERK_GEO [<geometry axis>]

With <geometry axis> = 0, 1, 2

Orientation axes

The maximum jerk when manually traversing orientation axes can be specified for each channel via the machine data:

MD21158 $MC_JOG_JERK_ORI [<orientation axis>]

For MD21158 to take effect, the channel-specific jerk limitation for the manual traversing of orientation axes must be enabled via the following machine data:

MD21159 $MC_JOG_JERK_ORI_ENABLE == TRUE

Reference:
Function Manual, Basic Functions; Acceleration (B2)

Alarms

Alarm 20062, "Axis already active", is triggered in the case that a geometry axis/orientation axis is manually traversed under the following conditions:

- The axis is already being traversed in JOG mode via the axial PLC interface.
- A frame for a rotated coordinate system is already active and another geometry axis in this coordinate system is traversed in JOG mode with the traversing keys.

If the axis is not defined as a geometry axis, alarm 20060, "Axis cannot be traversed as a geometry axis", is output if you attempt to traverse it as a geometry axis in JOG mode.

3.11.2 Spindle manual travel

Spindle manual travel

Spindles can also be traversed manually in JOG mode. Essentially, the same conditions apply as for manual travel of axes. Spindles can be traversed in JOG mode using the traversing keys continuously or incrementally, in jog or continuous mode, or using the handwheel. The function is selected and activated via the axis-/spindle-specific PLC interface in the same way as for the machine axes. The axis-specific machine data also apply to the spindles.

Spindle mode

Spindle manual travel is possible in positioning mode (spindle is in position control) or in open-loop control mode.

JOG velocity

The velocity used for spindle manual travel can be defined as follows:

- Using general setting data
  SD41200 $SN_JOG_SPIND_SET_VELO (speed of spindle in JOG mode), which is valid for all spindles

or

- Using machine data
  MD32020 $_MA_JOG_VELO (JOG axis velocity)

  However, the machine data is only effective if
  SD41110 $SN_JOG_SET_VELO (axis velocity in JOG) = 0.

The maximum speeds for the active gear stage also apply when spindles are traversed in JOG mode.

References:
Function Manual Basic Functions; Spindles (S1)

Velocity override

The spindle-override-switch JOG velocity is active for spindles.

JOG acceleration

As a spindle often uses many gear stages in speed-control and position-control modes, the acceleration associated with the current gear stage is always applied to the spindle in JOG mode.

References:
Function Manual Basic Functions; Spindles (S1)

PLC interface signals

In the case of spindle manual travel, the PLC interface signals between the NCK and PLC have the same effect as for machine axes.

Interface signals
DB31, ... DBX60.7 or DBX60.6 (position reached with fine or coarse exact stop)
are only set if the spindle is in position control.

In the case of interface signals that are only spindle-specific, while the spindles are traversing in JOG the following should be noted:

- The following PLC interface signals to the spindle have no effect:
  - DB31, ... DBX17.6 (invert M3/M4)
  - DB31, ... DBX18.6/7 (oscillation rotation direction right/left)
  - DB31, ... DBX18.5 (oscillation enable)
  - DB31, ... DBX16.7 (delete S value)
- The following PLC interface signals from the spindle are not set:
  - DB31, ... DBX83.7 (clockwise actual direction of rotation)
  - DB31, ... DBX83.5 (spindle in set range)

3.11.3 Monitoring functions

Limitations

The following limitations are active for manual and handwheel travel:

- Working-area limitation (axis must be referenced)
- Software limit switches 1 and 2 (axis must be referenced)
- Hardware limit switches

The control ensures that the traversing movement is aborted as soon as the first valid limitation has been reached. Velocity control ensures that deceleration is initiated early enough for the axis to stop exactly at the limit position (e.g. software limit switch). Only when the hardware limit switch is triggered does the axis stop abruptly with "rapid stop".

Alarms are triggered when the various limitations are reached (alarms 16016, 16017, 16020, 16021). The control automatically prevents further movement in this direction. The traversing keys and the handwheel have no effect in this direction.

Note

The software limit switches and working-area limitations are only active if the axis has first been referenced.

If a work offset (DRF offset) via handwheel is active for axes, the software limit switches for these axes are monitored during the main run in JOG mode. This means that the jerk limitation has no effect when the software limit switches are approached. After acceleration in accordance with MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration) the velocity is reduced at the software limit switch.

For further information on working area limitations and hardware and software limit switches, see:
References:
Function Manual, Axis Monitoring, Protection Zones (A3)

Retract axis

The axis can be retracted from a limit position by moving it in the opposite direction.

Note

Machine manufacturer

The function for retracting an axis that has approached the limit position depends on the machine manufacturer. Please refer to the machine manufacturer's documentation!

Maximum velocity and acceleration

The velocity and acceleration used during manual travel are defined by the startup engineer for specific axes using machine data. The control limits the values acting on the axes to the maximum velocity and acceleration specifications.

References:
Function Manual Basic Functions; Velocities, Setpoint/Actual Value Systems, Closed-Loop Controls (G2)
Function Manual Basic Functions; Acceleration (B2)

3.11.4 Other

Mode change from JOG to AUTO or from JOG to MDI

It is only possible to switch operating modes from JOG to AUTO or MDI if all axes in the channel have reached "coarse exact stop".

References:
Function Manual, Basic Functions; Mode Group, Channel, Program Operation, Reset Response (K1)

Rotational feedrate active in JOG

In JOG mode, it is also possible to traverse an axis manually at a revolutional feedrate (as for `G95`) specific to the current speed of the master spindle.

This is activated using the setting data:

SD41100$SN_JOG_REV_IS_ACTIVE (JOG: Revolutional/linear feedrate)

The feedrate value (in mm/rev) used can be defined as follows:

- with the general setting data:

  SD41120 $SN_JOG_REV_SET_VELO (revolutional feed of axes in JOG)
- using the axial machine data:

  MD32050 $MA_JOG_REV_VELO (revolutional feed rate for JOG mode)

  or for rapid traverse override:

  MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feedrate for JOG with rapid traverse override), if SD41120 = 0.

If a master spindle has not been defined and the axis is to be traversed in JOG at a revolutional feedrate, alarm 20055 is output (alarm 20065 for geometry axes).

Transverse axes

If a geometry axis is defined as transverse axis:
MD20100 $MC_DIAMETER_AX_DEF (geometry axes with transverse axis function)
and radius programming has been selected, when traversing in JOG, the following features should be carefully observed:

- Continuous travel:

  There are no differences when a transverse axis is traversed continuously.
- Incremental travel:

  Only **half the distance** of the selected increment size is traversed. For example, with INC10 the axis only traverses 5 increments when the traversing key is pressed.
- Traversing with the handwheel:

  As for incremental travel, with the handwheel only half the distance is traversed per handwheel pulse.

References:
Function Manual Basic Functions; Transverse Axes (P1)

3.12 Data lists

3.12.1 Machine data

3.12.1.1 General machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB[n]	Machine axis name
10720	OPERATING_MODE_DEFAULT	Setting of the operating mode after Power On
10721	OPERATING_MODE_EXTENDED	Extended setting of the operating mode after Power On
10735	JOG_MODE_MASK	Settings for JOG mode
11300	JOG_INC_MODE_LEVELTRIGGRD	INC and REF in inching mode
11310	HANDWH_REVERSE	Defines movement in the opposite direction
11320	HANDWH_IMP_PER_LATCH[n]	Handwheel pulses per locking position
11324	HANDWH_VDI_REPRESENTATION	Coding of the handwheel number (NCK/PLC interface)
11330	JOG_INCR_SIZE_TAB[n]	Increment size for INC/handwheel
11340	ENC_HANDWHEEL_SEGMENT_NR	Third handwheel: Bus segment
11342	ENC_HANDWHEEL_MODULE_NR	Third handwheel: Logical drive number
11344	ENC_HANDWHEEL_INPUT_NR	Third handwheel: Encoder interface
11346	HANDWH_TRUE_DISTANCE	Handwheel path or velocity specification
11350	HANDWHEEL_SEGMENT[n]	Handwheel segment

Number	Identifier: $MN_	Description
11351	HANDWHEEL_MODULE[n]	Handwheel module
11352	HANDWHEEL_INPUT[n]	Handwheel connection
11353	HANDWHEEL_LOGIC_ADDRESS[n]	Logical handwheel slot address (STEP 7)
17900	VDI_FUNCTION_MASK	Function mask for VDI signals

3.12.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20060	AXCONF_GEOAX_NAME_TAB	Geometry axis in channel
20100	DIAMETER_AX_DEF	Geometry axes with transverse axis functions
20110	RESET_MODE_MASK	Initial setting after reset / program end
20150	GCODE_RESET_VALUES	Reset position of the G groups
20151	GCODE_RESET_MODE	Reset behavior of the G groups
20360	TOOL_PARAMETER_DEF_MASK	Definition of the tool parameters
20620	HANDWH_GEOAX_MAX_INCR_SIZE	Limitation of the geometry axes
20622	HANDWH_GEOAX_MAX_INCR_VSIZE	Path-velocity override
20624	HANDWH_CHAN_STOP_COND	Definition of the behavior of traveling with handwheel, channel-specific
20700	REFP_NC_START_LOCK	NC start disable without reference point
21150	JOG_VELO_RAPID_ORI	Conventional rapid traverse for orientation axes
21158	JOG_JERK_ORI	Maximum jerk when manually traversing orientation axes
21159	JOG_JERK_ORI_ENABLE	Jerk limitation for manual travel of orientation axes enabled
21165	JOG_VELO_GEO	Conventional speed for geometry axes
21166	JOG_ACCEL_GEO	Maximum acceleration when manually traversing geometry axes
21168	JOG_JERK_GEO	Maximum jerk when manually traversing geometry axes

3.12.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30450	IS_CONCURRENT_POS_AX	Default setting at reset: Neutral axis or channel axis
30600	FIX_POINT_POS[n]	Fixed point positions of the axis
30610	NUM_FIX_POINT_POS	Number of fixed point positions of an axis
31090	JOG_INCR_WEIGHT	Weighting of an increment for INC/handwheel
32000	MAX_AX_VELO	Maximum axis velocity
32010	JOG_VELO_RAPID	Rapid traverse in jog mode

Number	Identifier: $MA_	Description
32020	JOG_VELO	Conventional axis velocity
32040	JOG_REV_VELO_RAPID	Revolutional feedrate in JOG mode with rapid traverse override
32050	JOG_REV_VELO	Revolutional feedrate for JOG
32060	POS_AX_VELO	Reset position for positioning-axis velocity
32080	HANDWH_MAX_INCR_SIZE	Limitation of the selected increment size
32082	HANDWH_MAX_INCR_VELO_SIZE	Limitation of the selected increment for velocity override
32084	HANDWH_STOP_COND	Behavior, handwheel travel
32090	HANDWH_VELO_OVERLAY_FACTOR	Ratio of JOG velocity to handwheel velocity (with DRF)
32300	MAX_AX_ACCEL	Maximum axis acceleration
32301	JOG_MAX_ACCEL	Maximum axial acceleration for JOG movements
32420	JOG_AND_POS_JERK_ENABLE	Initial setting for axial jerk limitation
32430	JOG_AND_POS_MAX_JERK	Maximum axial jerk for single axis movements
32431	MAX_AX_JERK	Maximum axial jerk for path movements
32436	JOG_MAX_JERK	Maximum axial jerk for JOG movements
34210	ENC_REFP_STATE	Measuring system status
35130	GEAR_STEP_MAX_VELO_LIMIT[n]	Maximum velocity for gear stage

3.12.2 Setting data

3.12.2.1 General setting data

Number	Identifier: $SN_	Description
41010	JOG_VAR_INCR_SIZE	Size of variable increment for INC/handwheel
41050	JOG_CONT_MODE_LEVELTRIGGRD	JOG continuous in inching mode
41100	JOG_REV_IS_ACTIVE	Revolutional feedrate in JOG mode active
41110	JOG_SET_VELO	JOG velocity for linear axes (for `G94`)
41120	JOG_REV_SET_VELO	JOG velocity (for `G95`)
41130	JOG_ROT_AX_SET_VELO	JOG velocity for rotary axes
41200	JOG_SPIND_SET_VELO	JOG velocity for the spindle

3.12.3 Signals

3.12.3.1 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Handwheel 1 is operated	DB10.DBB68	DB2700.DBB12
Handwheel 2 is operated	DB10.DBB69	DB2700.DBB13
Handwheel 3 is operated	DB10.DBB70	-
Handwheel 4 is operated	DB10.DBB242	-
Handwheel 5 is operated	DB10.DBB243	-
Handwheel 6 is operated	DB10.DBB244	-
Ethernet handwheel is stationary	DB10.DBB245	-

3.12.3.2 Signals to mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode Group1: JOG mode	DB11.DBX0.2	DB3000.DBX0.2
Mode group2: JOG mode	DB11.DBX20.2	-
Mode group3: JOG mode	DB11.DBX40.2	-
Mode group4: JOG mode	DB11.DBX60.2	-
Mode group5: JOG mode	DB11.DBX80.2	-
Mode group6: JOG mode	DB11.DBX100.2	-
Mode group7: JOG mode	DB11.DBX120.2	-
Mode group8: JOG mode	DB11.DBX140.2	-
Mode group9: JOG mode	DB11.DBX160.2	-
Mode group10: JOG mode	DB11.DBX180.2	-

3.12.3.3 Signals from mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode Group1: Active mode JOG	DB11.DBX6.2	DB3100.DBX0.2
Mode group2: Active mode JOG	DB11.DBX26.2	-
Mode group3: Active mode JOG	DB11.DBX46.2	-
Mode group4: Active mode JOG	DB11.DBX66.2	-
Mode group5: Active mode JOG	DB11.DBX86.2	-
Mode group6: Active mode JOG	DB11.DBX106.2	-
Mode group7: Active mode JOG	DB11.DBX126.2	-
Mode group8: Active mode JOG	DB11.DBX146.2	-

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode group9: Active mode JOG	DB11.DBX166.2	-
Mode group10: Active mode JOG	DB11.DBX186.2	-

3.12.3.4 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate DRF	DB21, ...DBX0.3	DB3200.DBX0.3
Geometry axis 1: Activate handwheel	DB21, ...DBX12.0-2	DB3200.DBX1000.0-2
Geometry axis 1: Traversing key lock	DB21, ...DBX12.4	DB3200.DBX1000.4
Geometry axis 1: Rapid traverse override	DB21, ...DBX12.5	DB3200.DBX1000.5
Geometry axis 1: Traversing keys minus/plus	DB21, ...DBX12.6-7	DB3200.DBX1000.6-7
Geometry axis 1: Machine function 1 INC ... Var. INC	DB21, ...DBX13.0-5	DB3200.DBX1001.0-5
Geometry axis 1: Invert handwheel direction of rotation	DB21, ...DBX15.0	DB3200.DBX1003.0
Geometry axis 2: Activate handwheel	DB21, ...DBX16.0-2	DB3200.DBX1004.0-2
Geometry axis 2: Traversing key lock	DB21, ...DBX16.4	DB3200.DBX1004.4
Geometry axis 2: Rapid traverse override	DB21, ...DBX16.5	DB3200.DBX1004.5
Geometry axis 2: Traversing keys minus/plus	DB21, ...DBX16.6-7	DB3200.DBX1004.6-7
Geometry axis 2: Machine function 1 INC ... Var. INC	DB21, ...DBX17.0-5	DB3200.DBX1005.0-5
Geometry axis 2: Invert handwheel direction of rotation	DB21, ...DBX19.0	DB3200.DBX1007.0
Geometry axis 3: Activate handwheel	DB21, ...DBX20.0-2	DB3200.DBX1008.0-2
Geometry axis 3: Traversing key lock	DB21, ...DBX20.4	DB3200.DBX1008.4
Geometry axis 3: Rapid traverse override	DB21, ...DBX20.5	DB3200.DBX1008.5
Geometry axis 3: Traversing keys minus/plus	DB21, ...DBX20.6-7	DB3200.DBX1008.6-7
Geometry axis 3: Machine function 1 INC ... Var. INC	DB21, ...DBX21.0-5	DB3200.DBX1009.0-5
Geometry axis 3: Invert handwheel direction of rotation	DB21,... .DBX23.0	DB3200.DBX1011.0
Activate contour handwheel	DB21, ...DBX30.0-2	DB3200.DBX14.0-2
Contour handwheel simulation on	DB21, ...DBX30.3	DB3200.DBX14.3
Contour handwheel simulation negative direction	DB21, ...DBX30.4	DB3200.DBX14.4
Invert contour handwheel direction of rotation	DB21, ...DBX31.5	DB3200.DBX15.5
Orientation axis 1: Activate handwheel	DB21, ...DBX320.0-2	-
Orientation axis 1: Traversing key lock	DB21, ...DBX320.4	-
Orientation axis 1: Rapid traverse override	DB21, ...DBX320.5	-
Orientation axis 1: Traversing keys minus/plus	DB21, ...DBX320.6-7	-
Orientation axis 1: Machine function 1 INC ... Var. INC	DB21, ...DBX321.0-5	-
Orientation axis 1: Invert handwheel direction of rotation	DB21, ...DBX323.0	-
Orientation axis 2: Activate handwheel	DB21, ...DBX324.0-2	-
Orientation axis 2: Traversing key lock	DB21, ...DBX324.4	-
Orientation axis 2: Rapid traverse override	DB21, ...DBX324.5	-

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Orientation axis 2: Traversing keys minus/plus	DB21, ...DBX324.6-7	-
Orientation axis 2: Machine function 1 INC ... Var. INC	DB21, ...DBX325.0-5	-
Orientation axis 2: Invert handwheel direction of rotation	DB21, ...DBX327.0	-
Orientation axis 3: Activate handwheel	DB21, ...DBX328.0-2	-
Orientation axis 3: Traversing key lock	DB21, ...DBX328.4	-
Orientation axis 3: Rapid traverse override	DB21, ...DBX328.5	-
Orientation axis 3: Traversing keys minus/plus	DB21, ...DBX328.6-7	-
Orientation axis 3: Machine function 1 INC ... Var. INC	DB21, ...DBX329.0-5	-
Orientation axis 3: Invert handwheel direction of rotation	DB21, ...DBX331.0	-

3.12.3.5 Signals from channel

Signal name		SINUMERIK 840D sl	SINUMERIK 828D
DRF selected		DB21, ...DBX24.3	DB1700.DBX0.3
Handwheel override active (path axes)		DB21, ...DBX33.3	DB3300.DBX1.3
Contour handwheel active		DB21, ...DBX37.0-2	DB3300.DBX5.0-2
Invert contour handwheel direction of rotation active		DB21, ...DBX39.5	DB3300.DBX7.5
Geometry axis 1	Handwheel active	DB21, ...DBX40.0-2	DB3300.DBX1000.0-1
	Traversing requests minus/plus	DB21, ...DBX40.4-5	DB3300.DBX1000.4-5
	Traversing command minus/plus	DB21, ...DBX40.6-7	DB3300.DBX1000.6-7
	Active machine function 1 INC ... Var. INC	DB21, ...DBX41.0-5	DB3300.DBX1001.0-5
	Handwheel direction of rotation inversion active	DB21, ...DBX43.0	DB3300.DBX1003.0
Geometry axis 2	Handwheel active	DB21, ...DBX46.0-2	DB3300.DBX1004.0-1
	Traversing requests minus/plus	DB21, ...DBX46.4-5	DB3300.DBX1004.4-5
	Traversing command minus/plus	DB21, ...DBX46.6-7	DB3300.DBX1004.6-7
	Active machine function 1 INC ... Var. INC	DB21, ...DBX47.0-5	DB3300.DBX1005.0-5
	Handwheel direction of rotation inversion active	DB21, ...DBX49.0	DB3300.DBX1007.0
Geometry axis 3	Handwheel active	DB21, ...DBX52.0-2	DB3300.DBX1008.0-1
	Traversing requests minus/plus	DB21, ...DBX52.4-5	DB3300.DBX1008.4-5
	Traversing command minus/plus	DB21, ...DBX52.6-7	DB3300.DBX1008.6-7
	Active machine function 1 INC ... Var. INC	DB21, ...DBX53.0-5	DB3300.DBX1009.0-5
	Handwheel direction of rotation inversion active	DB21, ...DBX55.0	DB3300.DBX1011.0

Signal name		SINUMERIK 840D sl	SINUMERIK 828D
Orientation axis 1	Handwheel active	DB21, ...DBX332.0-2	-
	Traversing request minus/plus	DB21, ...DBX332.4-5	-
	Traversing command minus/plus	DB21, ...DBX332.6-7	-
	Handwheel direction of rotation inversion active	DB21, ...DBX335.0	-
Orientation axis 2	Handwheel active	DB21, ...DBX336.0-2	-
	Traversing request minus/plus	DB21, ...DBX336.4-5	-
	Traversing command minus/plus	DB21, ...DBX336.6-7	-
	Handwheel direction of rotation inversion active	DB21, ...DBX339.0	-
Orientation axis 3	Handwheel active	DB21, ...DBX340.0-2	-
	Traversing request minus/plus	DB21, ...DBX340.4-5	-
	Traversing command minus/plus	DB21, ...DBX340.6-7	-
	Handwheel direction of rotation inversion active	DB21, ...DBX343.0	-
JOG retract active		DB21, ...DBX377.4	DB3300, DBX4005.4
Retraction data available		DB21, ...DBX377.5	DB3300, DBX4005.5

3.12.3.6 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate override	DB31, ...DBB0	DB380x.DBB0
Override active	DB31, ...DBX1.7	DB380x.DBX1.7
Delete distance-to-go/spindle reset	DB31, ...DBX2.2	DB380x.DBX2.2
Activate handwheel	DB31, ...DBX4.0-2	DB380x.DBX4.0-2
Traversing key lock	DB31, ...DBX4.4	DB380x.DBX4.4
Rapid traverse override	DB31, ...DBX4.5	DB380x.DBX4.5
Traversing keys minus/plus	DB31, ...DBX4.6-7	DB380x.DBX4.6-7
Machine function 1 INC ... Var. INC	DB31, ...DBX5.0-5	DB380x.DBX5.0-5
Invert handwheel direction of rotation	DB31, ...DBX7.0	DB380x.DBX7.0
JOG fixed point approach	DB31, ...DBX13.0-2	DB380x.DBX1001.0-2

3.12.3.7 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Position reached with coarse/fine exact stop	DB31, ...DBX60.6-7	DB390x.DBX0.6-7
Handwheel override active	DB31, ...DBX62.1	DB390x.DBX2.1
Handwheel active	DB31, ...DBX64.0-2	DB390x.DBX4.0-2
Traversing request minus/plus	DB31, ...DBX64.4-5	DB390x.DBX4.4-5

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Traversing command minus/plus	DB31, ...DBX64.6-7	DB390x.DBX4.6-7
Active machine function 1 INC ... Var. INC	DB31, ...DBX65.0-5	DB390x.DBX5-0-5
Position restored, measuring system 1/2	DB31, ...DBX71.4-5	DB390x.DBX11.4-5
JOG approach fixed point active	DB31, ...DBX75.0-2	DB390x.DBX1001.0-2
JOG approach fixed point reached	DB31, ...DBX75.3-5	DB390x.DBX1001.3-5
Handwheel direction of rotation inversion active	DB31, ...DBX67.0	DB390x.DBX7.0

3.12.4 System variable

3.12.4.1 System variable

Identifier	Description
$AA_POSRES	Axis state "Position restored"

3.12.5 OPI variable

3.12.5.1 OPI variable

Identifier	Description
retractState, bit 0	Retraction data available
retractState, bit 1	JOG retract active
retractState, bits 2 - 3	Retraction axis
aaPosRes	Axis state "Position restored".

K3: Compensations 4

4.1 Introduction

Accuracy errors

The accuracy of machine tools is impaired as a result of deviations from the ideal geometry, power transmission faults and measuring system errors. Temperature differences and mechanical forces often result in great reductions in precision when large workpieces are machined.

Compensation functions

Some of these deviations can usually be measured during commissioning and then compensated for during operation on the basis of values read by the positional actual-value encoder and other sensory devices. State-of-the-art CNC controls have compensation functions that are active on an axis for axis basis.

The following compensation functions are available:

- Temperature compensation
- Backlash compensation
  - Mechanical backlash compensation
  - Dynamic backlash compensation
- Interpolatory compensation
  - Compensation of leadscrew errors and measuring system errors
  - Compensation of sag and angularity errors
- Dynamic feedforward control (following error compensation)
- Friction compensation (quadrant error compensation)
  - Conventional friction compensation
- Electronic counterweight

Parameterization

These compensation functions can be set for each machine individually with axis-specific machine data.

Activation

The compensations are active in all operating modes of the control as soon as the input data are available. Any compensations that require the position actual value are not activated until the axis reaches the reference point.

Position display

The normal actual-value and setpoint position displays ignore the compensation values and show the position values of an ideal machine. The compensation values are output in the "Diagnosis" operating area of the "Axis/Spindle Service" window.

4.2 Temperature compensation

4.2.1 Description of functions

Deformation due to temperature effects

Heat generated by the drive equipment or high ambient temperatures (e.g. caused by sunlight, drafts) cause the machine base and parts of the machinery to expand. This expansion depends, among other things, on the temperature and on the thermal conductivity of the machine parts.

Effects

Due to the thermal expansion of the machinery, the actual positions of the axes change depending on temperature. This has a negative impact on the precision of the workpieces being machined.

Temperature compensation

By activating the "temperature compensation" function, actual value changes due to temperature effects can be compensated on an axis-by-axis basis.

Sensor equipment

To provide effective temperature compensation, a number of temperature sensors for acquiring a temperature profile are needed in addition to the actual position data from existing encoders.

Since temperature-dependent changes occur relatively slowly, the PLC can acquire and preprocess the temperature profile in a minutes cycle, for example.

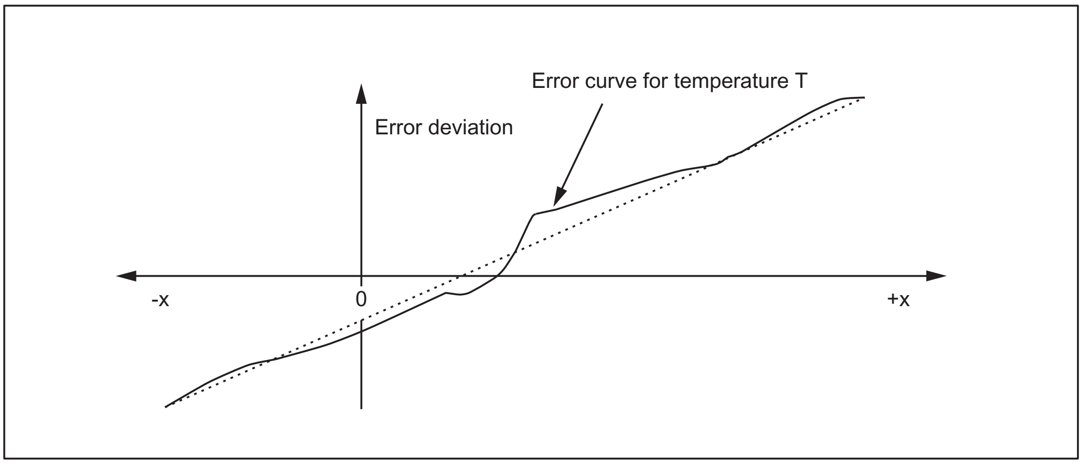
Error curves

In order to implement temperature compensation, the actual-value offsets over the positioning range of the axis must be measured at a given temperature (T) and plotted. This produces an error curve for this temperature value. Error curves must be produced for different temperatures.

Error curve characteristic

If an axis position reference point P_0 is selected, an offset in the reference point (corresponds to the "position-independent component" of the temperature compensation) can be observed as the temperature changes, and because of the change in length an additional offset in the other position points, which increases with the distance to the reference point (corresponds to the "position-dependent component" of the temperature compensation).

The error curve for a given temperature T can generally be represented with sufficient accuracy by a straight line with a temperature dependent gradient and reference position.



Compensation equation

The compensation value ΔK_x is calculated on the basis of current actual position P_x of this axis and temperature T according to the following equation:

$\Delta K_x = K_0 (T) + \tan\beta (T) * (P_x - P_0)$

The meaning is as follows:

ΔK_x:	Temperature compensation value of axis at position P_x
K_0:	Position-independent temperature compensation value of axis
P_x:	Actual position of axis
P_0	Reference position of axis
tanβ	Coefficient for the position-dependent temperature compensation (corresponds to the gradient of the approximated error line)

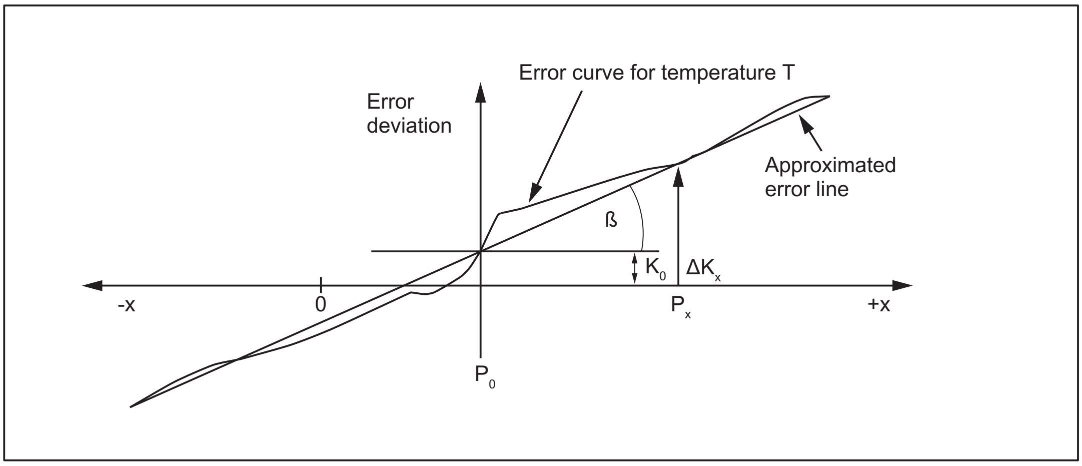


Figure 4-1 Approximated error line for temperature compensation

Activation

The following conditions must be fulfilled so that the temperature compensation can be activated:

1. The compensation type is selected (MD32750, see "Commissioning (Page 227)").
2. The parameters for the compensation type are defined (see "Commissioning (Page 227)").
3. The axis is referenced:

   DB31, ... DBX60.4 or 60.5 =1 (referenced/synchronized 1 or 2 respectivelly)

As soon as these conditions are fulfilled, the temperature compensation value for the position actual value is added to the setpoint in all modes and the machine axis traverses through this distance. If the compensation value ΔK_x is positive, the axis moves in the negative direction.

If the reference position is then lost, e.g. because the encoder frequency has been exceeded (DB31, ... DBX60.4 or 60.5 = 0), compensation processing is deactivated.

Clock cycle

The compensation values are determined in the interpolation cycle.

Display

The total compensation value calculated from the temperature and sag compensation functions belonging to the actual position is output in the "Diagnosis" operating area of the "Axis/Spindle Service" window.

Parameter adaptation for temperature changes

Since the approximated error line applies only to the instantaneous temperature value, the parameters of the error lines that are newly generated when the temperature rises or falls must be sent to the NCK again. Only in this way can expansion due to heat always be correctly compensated.

When temperature T changes, the parameters which are temperature-dependent, i.e. (K_0, tanβ and P_0) also change and can thus always be overwritten by the PLC or by means of a synchronized action.

It is thus possible for the machine-tool manufacturer to emulate the mathematical and technological relationship between the axis positions and temperature values via the PLC user program and thus calculate the various parameters for the temperature compensation. The temperature parameters are transferred to the NCK using the variable services (FB2 (GET) "Read data" and FB3 (PUT) "Write data").

For more information on handling and supplying FB2 and FB3 see:
Reference:
Function Manual Basic Functions; Basic PLC Program (P3)

Smooth the compensation value

To prevent overloading of the machine or tripping of monitoring functions in response to step changes in the temperature compensation parameters, the compensation values are distributed over several IPO cycles by an internal control function as soon as they exceed the maximum compensation value specified for each IPO cycle (MD32760, see "Commissioning (Page 227)").

4.2.2 Commissioning

Temperature-dependent parameters

Error curves for different temperatures can be defined for each axis. For each error curve the following parameters must be determined and then entered in the setting data:

- Position-independent temperature compensation value K_0:

  SD43900 $SA_TEMP_COMP_ABS_VALUE
- Reference position P_0 for position-dependent temperature compensation:

  SD43920 $SA_TEMP_COMP_REF_POSITION
- Gradient tanβ for position-dependent temperature compensation:

  SD43910 $SA_TEMP_COMP_SLOPE

Temperature compensation type and activation

The temperature compensation type is selected and the temperature compensation activated using the axis-specific machine data:

MD32750 $MA_TEMP_COMP_TYPE (temperature compensation type)

Bit	Value	Meaning	Associated parameters
0	Position independent temperature compensation		SD43900
	0	Not active	
	1	Active	
1	Position-dependent temperature compensation		SD43920, SD43910
	0	Not active	
	1	Active	
2	Temperature compensation in tool direction		MD20390 $MC_TOOL_TEMP_COMP_ON (Activate temperature compensation tool length)
	0	Not active	
	1	Active	

Maximum compensation value per IPO clock cycle

The maximum possible compensation value per IPO cycle, i.e. the maximum distance that can be traversed in an IPO cycle as a result of the temperature compensation, is limited using machine data:

MD32760 $MA_COMP_ADD_VELO_FACTOR (velocity increase as a result of compensation)

The specified value acts as a factor and is referred to the maximum axis velocity (MD32000 $MA_MAX_AX_VELO).

MD32760 also limits the maximum gradient of the error line (tan ß) of the temperature compensation.

4.2.3 Example

4.2.3.1 Commissioning the temperature compensation for the Z axis of a lathe

Commissioning of temperature compensation is described below using the example of a Z axis on a lathe.

Determining the error characteristic of the Z axis

In order to determine the temperature-dependent error characteristic of the Z axis, proceed as follows:

- Uniform temperature increase by traversing the axis across the whole Z axis traversing range (in the example: from 500 mm to 1500 mm)
- Measuring the axis position in increments of 100 mm
- Measuring the actual temperature at the leadscrew
- Executing a traversing measuring cycle every 20 minutes

The mathematical and technological relationships and the resulting parameters for temperature compensation are derived from the recorded data. The calculated deviation errors for a specific temperature, which refer to the actual position of the Z axis displayed by the NC, are represented in graphic form in the diagram below.

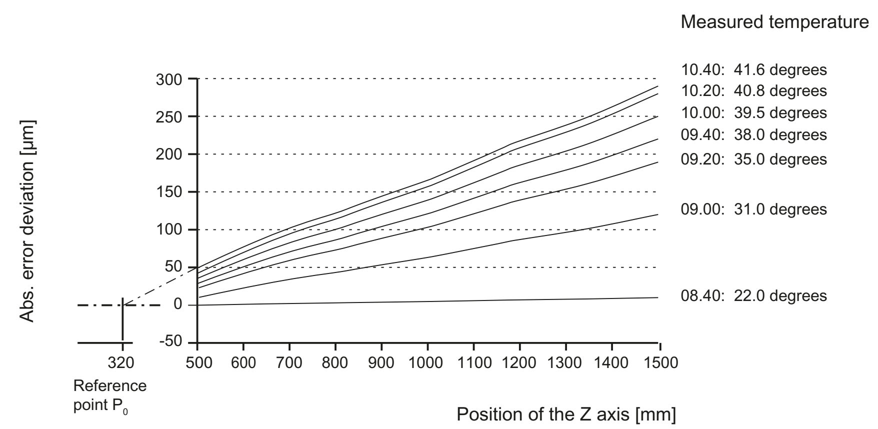


Figure 4-2 Error curves determined for the Z axis

Specifying parameters

The temperature compensation parameters must now be determined on the basis of the measurement results (see diagram above).

Reference position P_0

As the diagram above illustrates, there are basically two methods of parameterizing reference position P_0:

1. $P_0 = 0$ with position-independent temperature compensation value $K_0 \neq 0$
2. $P_0 \neq 0$ with position-independent temperature compensation value $K_0 = 0$

In this case, version 2 is chosen, which means that the position-independent temperature compensation value is always 0. The temperature compensation value therefore only consists of the position-dependent component.

The following parameters are obtained:

- MD32750 $MA_TEMP_COMP_TYPE = 2

  (only position-dependent temperature compensation active)
- P_0 = 320 mm → SD43920 $SA_TEMP_COMP_REF_POSITION = 320

  (reference position for position-dependent temperature compensation)

Coefficient tanβ (T)

In order to determine the dependency of coefficient tanβ of the position-dependent temperature compensation on the temperature, the error curve gradient is plotted against the measured temperature:

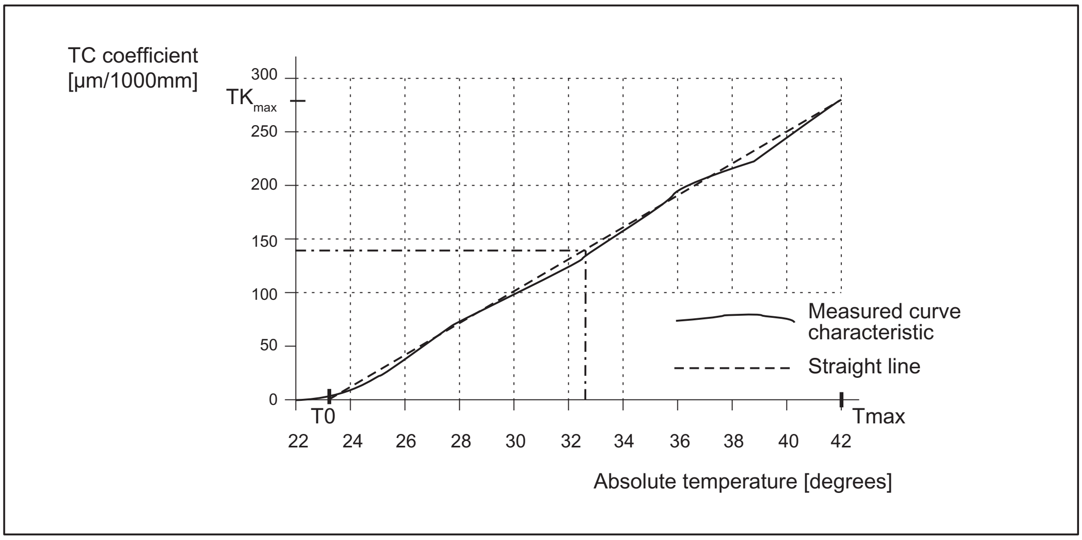


Figure 4-3 Characteristic of coefficient tanβ as a function of measured temperature T

With the appropriate linearization, coefficient tanβ depends on T as follows:

$\tan\beta(T) = (T - T_0) * TK_{max} * 10^{-6} / (T_{max} - T_0)$

with

T_0 = temperature at which position-dependent error = 0; [degrees]

T_{max} = maximum measured temperature; [degrees]

TK_{max} = temperature coefficient at T_{max}; [µm/1000 mm]

Therefore, based on the values from the above diagram:

T_0 = 23°

T_{max} = 42°

TK_{max} = 270 µm/1000 mm

and tanß (T) is therefore:

tanβ(T) = (T - 23) [degrees] * 270 [µm/1000 mm] * 10^{-6} / (42 - 23) [degrees]
= (T - 23) [degrees] * 14.21 [µm/1000 mm] * 10^{-6}

Example:

At a temperature of T = 32.3 degrees, therefore: tanβ = 0.000132

PLC user program

The formula given above must be used in the PLC user program to calculate the coefficient tanβ (T) which corresponds to the measured temperature; this must then be written to the following NCK setting data:

SD43910 $SA_TEMP_COMP_SLOPE (gradient for position-dependent temperature compensation)

According to the example above:

SD43910 $SA_TEMP_COMP_SLOPE = 0.000132

4.3 Backlash compensation

4.3.1 Mechanical backlash compensation

4.3.1.1 Description of functions

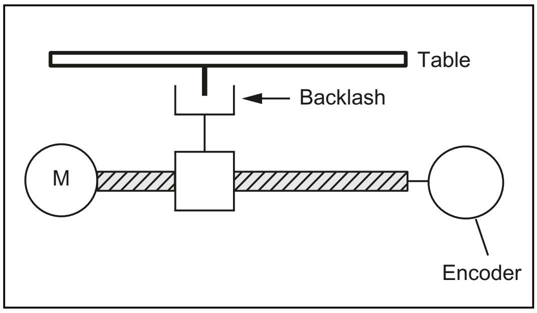
Mechanical backlash

When force is transmitted between a moving machine part and its drive (e.g. ball screw), there is normally a small amount of backlash because adjusting mechanical parts so that they are completely free of backlash would result in too much wear and tear on the machine. Thus, backlash (play) can occur between the machine component and the measuring system.

Effects

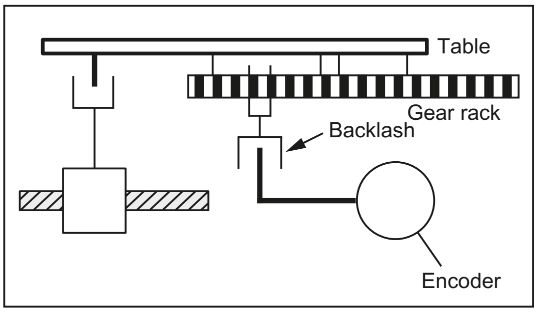
In the case of axes/spindles with indirect measuring systems, mechanical backlash falsifies the traversing path. For example, when the direction of movement is reversed, an axis will travel too much or too little by the amount of the backlash.

Positive backlash (normal case)



The encoder leads the machine part (e.g. table). Since the actual position acquired by the encoder also leads the real actual position of the table, the table travels too short a distance.

Negative backlash



The encoder lags behind the machine part (e.g. table). The table travels too far as the actual position of the table leads the actual position detected by the encoder.

Compensation

To compensate for the mechanical backlash, the axis-specific actual value is corrected by the axis-specific backlash amount specified when commissioning (MD32450, see "Commissioning (Page 233)") every time the axis/spindle changes direction.

Activation

The mechanical backlash compensation is always active in all operating modes after reference point approach.

Display

The compensation value that belongs to the current actual position is displayed in the "Diagnostics" operator area of the "Axis/Spindle Service" window as the total compensation calculated from LEC, mechanical and dynamic backlash compensation.

4.3.1.2 Commissioning

Backlash

The mechanical backlash is measured at high velocity with a measuring system attached to the machine table.

The determined offset value is entered in the axis-specific machine data:

MD32450 $MA_BACKLASH (backlash)

For positive backlash (normal case), the compensation value should be entered as a positive value and for a negative backlash, as negative value.

After activation of the machine data (NewConfig), the measurement is repeated and the effect of the mechanical backlash compensation checked.

Note

The circularity test integrated in the HMI can be used to visualize the error.

2nd measuring system

If there is a 2nd measuring system for the axis/spindle, a backlash compensation must be entered for this too. As the second measuring system is mounted in a different way from the first measuring system, the backlash can be different from that of the first measuring system.

When the measuring system is switched over the associated compensation value is always automatically activated.

Weighting factor for backlash

The backlash can be weighted by a factor dependent on the particular parameter set:

MD32452 $MA_BACKLASH_FACTOR (backlash weighting factor)

Value range: 0.01 ... 100.0

Preset: 1.0

Application: e.g. compensation of gear-stage-dependent backlash.

Maximum tolerance for position actual value switchover

The user has the option of applying the backlash compensation value gradually in several increments when the relevant axis reverses direction. This prevents a setpoint step change on the axes from causing corresponding errors.

The contents of the following axis-specific machine data determine the increment with which the backlash compensation value (MD32450) is applied.

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

Please note that the backlash compensation is only fully calculated after <n> servo cycles (<n> = MD32450 / MD36500). An excessive time span can cause the triggering of a standstill monitoring alarm. If MD36500 > MD32450, then the compensation is performed in one servo cycle.

4.3.2 Dynamic backlash compensation

4.3.2.1 Description of functions

Dynamic backlash

A dynamic backlash can occur for machine types with sliding guides. Depending on the axial dynamic response (velocity, jerk, etc.) used to approach an end position, the machine slide reaches the programmed end position or stops earlier because of the static friction. The resulting position error is direction-symmetric.

Compensation

To compensate for the dynamic backlash, half the signed compensation value (MD32456Commissioning (Page 235), see "") is applied in accordance with the relevant traversing direction of the axis. Compensation value is applied as ramp.

Activation

The dynamic backlash compensation is activated by the PLC only in required the situations:

DB31, ... DBX25.0 (activate dynamic backlash compensation)

Note

The machine tool manufacturer specifies in the PLC user program the "required" situations for the activation of the dynamic backlash compensation. Such situations result with traversal of the axes with G1, in the JOG mode or with the handwheel.

The NC uses the following interface to return the required activation to the PLC:

DB31, ... DBX102.0 (active dynamic backlash compensation)

Requirement

The axes to be compensated must be homed.

Display

The compensation value that belongs to the current actual position is displayed in the "Diagnostics" operator area of the "Axis/Spindle Service" window as the total compensation calculated from LEC, mechanical and dynamic backlash compensation.

4.3.2.2 Commissioning

Compensation value for the dynamic backlash compensation

Once the mechanical backlash at highspeed has been determined by measurement and compensated using MD32450 $MA_BACKLASH (see "Commissioning (Page 233)" of the mechanical backlash compensation), the same measurement action is repeated for slower velocities. The backlash now measured represents the compensation value for the dynamic backlash compensation. The value is entered in the machine data:

MD32456 $MA_BACKLASH_DYN (compensation value for dynamic backlash compensation)

Limitation of the compensation value change

Compensation value for the dynamic backlash compensation is applied as ramp. The ramp is specified by the following machine data:

MD32457 $MA_BACKLASH_DYN_MAX_VELO (limitation of the dynamic backlash compensation value change)

The value is specified as percentage of the configured maximum axis velocity (MD32000 $MA_MAX_AX_VELO). The standard setting is 1%.

4.4 Interpolatory compensation

4.4.1 General properties

Function

The "Interpolatory compensation" function allows position-related dimensional deviations (for example, leadscrew and measuring system errors, sag and angularity errors) to be corrected.

The compensation values are measured during commissioning and stored in a table as a position-related value. During operation, the corresponding axis is compensated between interpolation points during linear interpolation.

Methods

Within "interpolatory compensation", a distinction is made between the two following compensation methods:

- Compensation of leadscrew errors and measuring system errors
- Compensation of sag and angularity errors

Many of the characteristics of these two compensation methods are identical and are therefore subsequently described for the two methods.

Terms

Important terms for "interpolatory compensation" are:

- Compensation value

  The difference between the axis position measured by the position actual-value encoder and the required programmed axis position (= axis position of the ideal machine). The compensation value is often also referred to as the correction value.

- Basic axis

  Axis whose setpoint position or actual position forms the basis for calculating a compensation value.

- Compensation axis

  Axis whose setpoint position or actual position is modified by a compensation value.

- Interpolation point

  A position of the basic axis and the corresponding compensation value of the compensation axis.

- Compensation table

  Table of interpolation points and compensation values (see below)

- Compensation relation

  Assignment of the basic axis and the corresponding compensation axis and the reference to the corresponding compensation table.

Compensation tables

Because the mentioned dimensional deviations directly affect the accuracy of workpiece machining, they must be compensated for by the relevant position-dependent compensation values. The compensation values are derived from measured error curves and entered in the control in the form of compensation tables during commissioning. A separate table must be created for each compensation relation.

Entering compensation tables

The size of the compensation table, i.e. the number of interpolation points, must first be defined in a machine data. After the next POWER ON, the compensation tables are generated by the NC and preassigned a value of "0".

The compensation values and additional table parameters are entered in the compensation tables using special system variables. Data can be loaded in two different ways:

- By starting an NC program with the parameter values.
- By transferring the compensation tables from an external computer to the control.

Note

Compensation tables can only be loaded if the corresponding compensation function is not active:

- MD32700 $MA_ENC_COMP_ENABLE (interpolatory compensation) = 0
- MD32710 $MA_CEC_ENABLE (enable sag compensation) = 0

These compensation values are not lost when the control is switched off because they are stored in the static user memory. They can be updated if necessary (e.g. following re-measuring because of machine aging).

Note

When changing machine data:

- MD18342 $MN_MM_CEC_MAX_POINTS
- MD38000 $MA_MM_ENC_COMP_MAX_POINTS

is formatted during the next system run-up of the static user memory (see Section "S7: Memory configuration (Page 755)").

Linear interpolation between interpolation points

The traversing path to be compensated delineated by the start and end positions is divided up into several (number depends on error curve shape) path segments of equal size (see diagram below). The actual positions that limit these sub-paths are referred to as "interpolation points" below. A compensation value must be entered for each interpolation point (actual position) during commissioning. The compensation value applied between two interpolation points is generated on the basis of **linear interpolation** using the compensation values for the adjacent interpolation points (i.e. adjacent interpolation points are linked along a line).

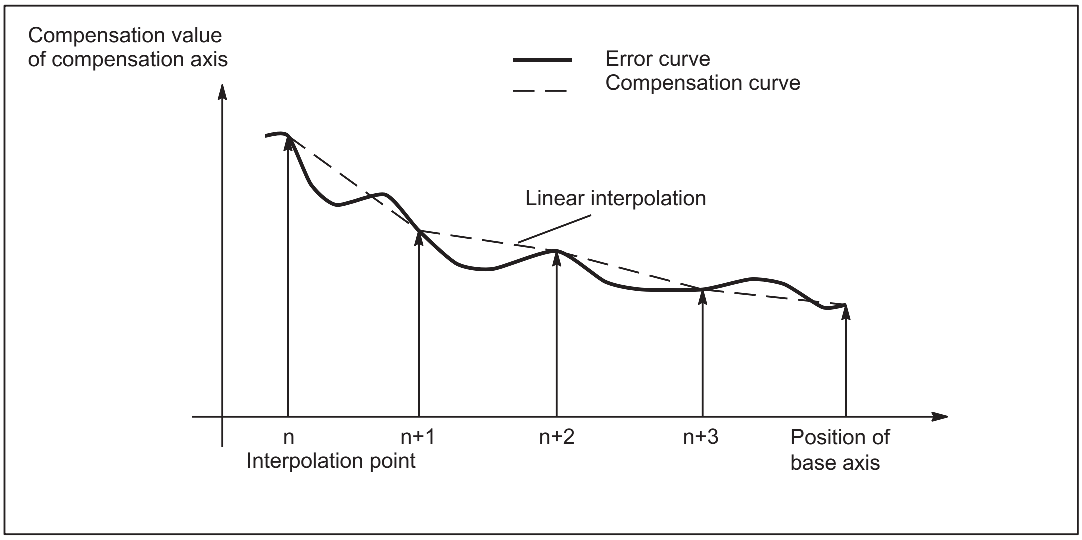


Figure 4-4 Linear interpolation between the interpolation points

Compensation value at reference point

The compensation table should be structured such that the compensation value at the reference point is "zero".

4.4.2 Compensation of leadscrew errors and measuring system errors

4.4.2.1 Measuring system error compensation (MSEC)

Leadscrew and measuring system errors

The measuring principle of "indirect measurement" on NC-controlled machines is based on the assumption that the lead of the ball screw is constant at any point within the traversing range, so that the actual position of the axis can be derived from the position of the drive spindle (ideal case). However, manufacturing tolerances result in dimensional deviations of varying degrees of severity on spindles (so-called leadscrew errors).

Added to this are the dimensional deviations (differences in reference division) caused by the measuring system as well as its mounting on the machine (so-called measuring system errors), plus any machine-dependent error sources.

Compensation

With "measuring system error compensation" (referred to below as MSEC), the base and compensation axes are always identical. It is therefore an axial compensation for which a definition of the base axis and compensation axis in the compensation table is not necessary.

Note

The leadscrew error compensation (LEC) is part of the measuring system error compensation.

The principle of the MSEC is to modify the axis-specific position actual value by the assigned compensation value in the interpolation cycle and to apply this value to the machine axis for immediate traversal. A positive compensation value causes the corresponding machine axis to move in the negative direction.

The magnitude of the compensation value is not limited and is not monitored. In order to avoid impermissibly high velocities and accelerations caused by compensation, small compensation values must be selected. Large compensation values can cause other axis monitoring functions to output alarms (e.g. contour monitoring, speed setpoint limitation).

If the axis to be compensated has a 2nd position measuring system, a separate compensation table must be created and activated for each measuring system. The correct table is automatically used when switching between measuring systems.

Preconditions / activation

The MSEC is only active until the following pre-conditions:

- The compensation values are stored in the static user memory and are active (after POWER ON).
- The function has been activated for the relevant machine axis:

  MD32700 $MA_ENC_COMP_ENABLE [<e>] = 1

with: <e> = Position measuring system

<e> = 0	Measuring system 1
<e> = 1	Measuring system 2

- The axis has been referenced:

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

As soon as these conditions have been fulfilled, the axis-specific actual value is modified by the compensation value in all modes and traversed by the machine axis immediately.

If the reference is then lost, e.g. because the encoder frequency has been exceeded (DB31, ... DBX60.4 or 60.5 = 0), compensation processing is deactivated.

4.4.2.2 Commissioning

Number of compensation interpolation points (MD38000)

For every machine axis and for every measuring system (if a 2nd measuring system is installed), the number of reserved interpolation points of the compensation table must be defined and the necessary memory reserved with the following machine data:

MD38000 $MA_MM_ENC_COMP_MAX_POINTS[<e>,<AXi>]

with: <e> = Position measuring system

<e> = 0	Measuring system 1
<e> = 1	Measuring system 2

<AXi> = Axis name (e.g. X1, Y1, Z1)

The required number of compensation interpolation points is calculated as follows:

$$\text{MM\_ENC\_COMP\_MAX\_POINTS [e,AXi]} = \frac{\text{\$AA\_ENC\_COMP\_MAX[e,AXi]} - \text{\$AA\_ENC\_COMP\_MIN[e,AXi]}}{\text{\$AA\_ENC\_COMP\_STEP[e,AXi]}} + 1$$

Measuring system-specific parameters of the compensation table

The position-related compensations as well as additional table parameters should be saved in the form of system variables for each machine axis as well as for each measuring system (if a 2nd measuring system is being used):

- **$AA_ENC_COMP[<e>,<N>,<AXi>]**

  (Compensation value for interpolation point N in the compensation table)

  <N> = interpolation point (axis position)

  For every individual interpolation point the compensation value must be entered in the table.

  <N> is limited by the maximum number of interpolation points of the particular compensation table (MD38000 $MA_MM_ENC_COMP_MAX_POINTS):

  0 ≤ N ≤ MD38000 -1

  The size of the compensation value is not limited.

  Note

  The first and last compensation values remain active over the entire traversing range; i.e. these values should be set to "0" if the compensation table does not cover the entire traversing range.

- **$AA_ENC_COMP_STEP[<e>,<AXi>] (distance between interpolation points)**

  The distance between interpolation points defines the distance between the compensation values in the relevant compensation table.

- **$AA_ENC_COMP_MIN[<e>,<AXi>] (initial position)**

  The initial position is the axis position at which the compensation table for the relevant axis begins (≙ interpolation point 0).

  The compensation value for the initial position is $AA_ENC_COMP[<e>,**0**,<AXi>)].

  The compensation value of interpolation point 0 is used for all positions smaller than the initial position (does not apply for tables with modulo function).

- **$AA_ENC_COMP_MAX[<e>,<AXi>] (end position)**

  The end position is the axis position at which the compensation table for the relevant axis ends (≙ interpolation point <k>).

  The compensation value for the end position is $AA_ENC_COMP[<e>,**<k>**,<AXi>)].

  The compensation value of interpolation point <k> is used for all positions larger than the end position (exception for table with modulo function).

  The following supplementary conditions apply to interpolation point <k>:

  – for k = MD38000 - 1:

    The compensation table is fully utilized!

  – for k < MD38000 - 1:

    The compensation table is not fully utilized. Compensation values entered in the table that are greater than k are ignored.

  – for k > MD38000 - 1:

    The compensation table is limited by a control function which reduces the end position. Compensation values that are greater than k are ignored.

- **$AA_ENC_COMP_IS_MODULO[<e>,<AXi>] (compensation with modulo function)**

  System variable to activate/deactivate the compensation with modulo function:

  – $AA_ENC_COMP_IS_MODULO[<e>,<AXi>] **= 0**: Compensation **without** modulo function

  – $AA_ENC_COMP_IS_MODULO[<e>,<AXi>] **= 1**: Compensation **with** modulo function

  When compensation with modulo function is activated, the compensation table is repeated cyclically, i.e. the compensation value at position $AA_ENC_COMP_MAX (≙ interpolation point $AA_ENC_COMP[<e>,<k>,<AXi>]) is immediately followed by the compensation value at position $AA_ENC_COMP_MIN (≙ interpolation point $AA_ENC_COMP[<e>,<0>,<AXi>]).

  For rotary axes with modulo 360° degrees it is therefore suitable to program 0° ($AA_ENC_COMP_MIN) as the initial position and 360° ($AA_ENC_COMP_MAX) as the end position.

  The compensation values entered for these two positions should be the same as otherwise the compensation value jumps from MAX to MIN at the transition point and vice versa.

CAUTION

When the compensation values are entered, it is important that all interpolation points within the defined range are assigned a compensation value (i.e. there should be no gaps). Otherwise, the compensation value that was left over from previous entries at these positions is used for these interpolation points.

Note

Table parameters containing position information are automatically converted when the system of units is changed (change from MD10240 $MN_SCALING_SYSTEM_IS_METRIC).

The position information is always interpreted in the actual system of units. Conversion must be implemented externally.

Automatic conversion of the position data can be configured as follows:

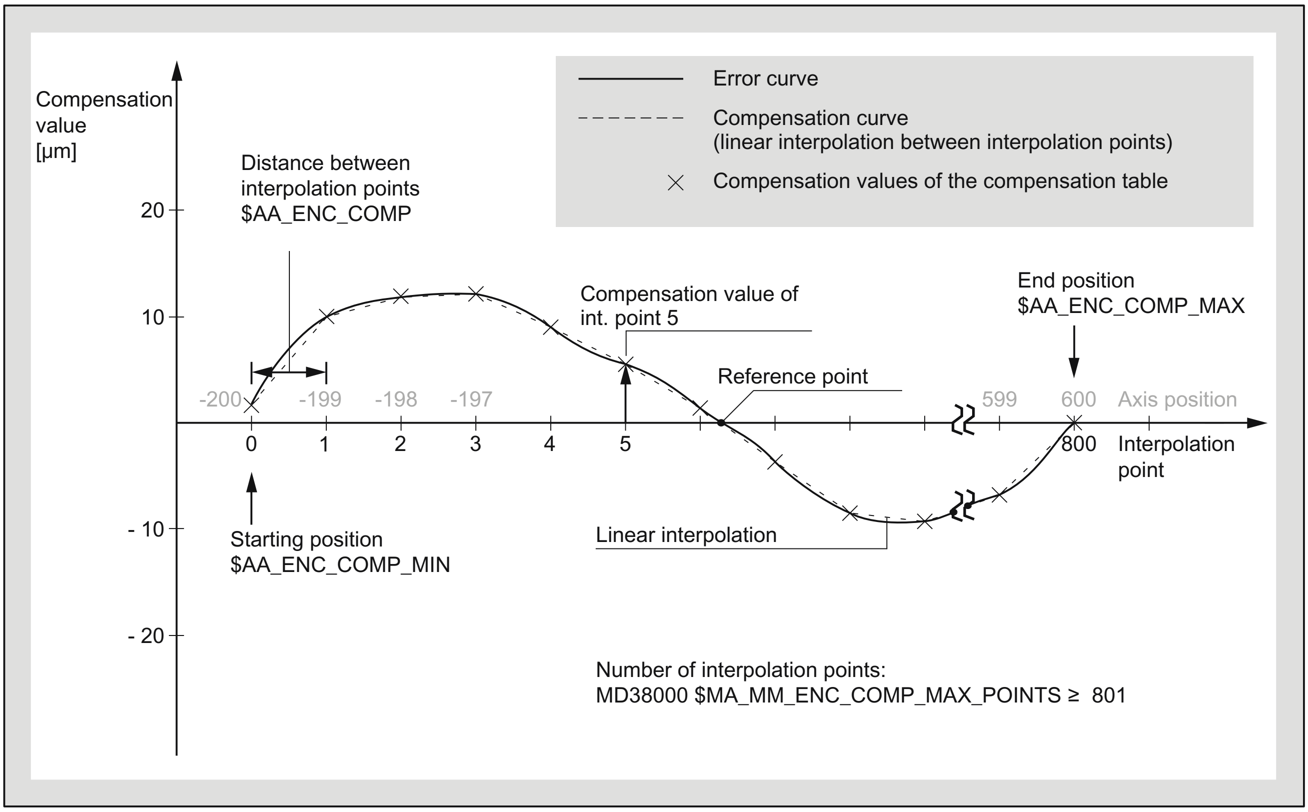
MD10260 $MN_CONVERT_SCALING_SYSTEM = 1

External conversion is no longer necessary.

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

4.4.2.3 Example

The following example shows compensation value inputs for machine axis X1.



Program code	Comment
`%_N_AX_EEC_INI`	
`CHANDATA(1)`	
`$AA_ENC_COMP[0,0,X1]=0.003`	; 1st compensation value (interpolation point 0): +3µm
`$AA_ENC_COMP[0,1,X1]=0.01`	; 2nd compensation value (interpolation point 1): +10µm
`$AA_ENC_COMP[0,2,X1]=0.012`	; 3rd compensation value (interpolation point 2): +12µm
`...`	
`$AA_ENC_COMP[0,800,X1]=-0.0`	; Last compensation value (interpolation point 800): 0µm
`$AA_ENC_COMP_STEP[0,X1]=1.0`	; Distance between interpolation points 1.0 mm
`$AA_ENC_COMP_MIN[0,X1]=-200.0`	; Compensation starts at -200.0 mm
`$AA_ENC_COMP_MAX[0,X1]=600.0`	; Compensation ends at +600.0 mm
`$AA_ENC_COMP_IS_MODULO[0,X1]=0`	; Compensation without modulo function
`M17`	

For this example, the configured number of interpolation points must be ≥ 801:
MD38000 $MA_MM_ENC_COMP_MAX_POINTS ≥ 801
The memory required in the static user memory is 6.4 KB (8 bytes per compensation value).

4.4.3 Compensation of sag and angularity errors

4.4.3.1 Description of functions

Sag errors

Weight can result in position-dependent displacement and inclination of moved parts since it can cause machine parts and their guides to sag.

Also large workpieces (e.g. cylinders) sag under their own weight.

Angularity errors

If moving axes are not positioned in exactly the required angle (e.g. perpendicular) with respect to one another, increasingly serious positioning errors will occur as the deviation from zero point becomes greater.

Compensation

In contrast to the MSEC, the base and compensation axes need not be identical for "Sag compensation" or "Angularity error compensation", requiring an axis assignment in every compensation table.

In order to compensate for sag of one axis (base axis) which results from its own weight, the absolute position of another axis (compensation axis) must be influenced. "Sag compensation" is therefore an **inter-axis compensation**.

As illustrated in the diagram below, the further the machining head moves in the negative Y1 axis direction, the more the cross-arm sags in the negative Z1 axis direction.

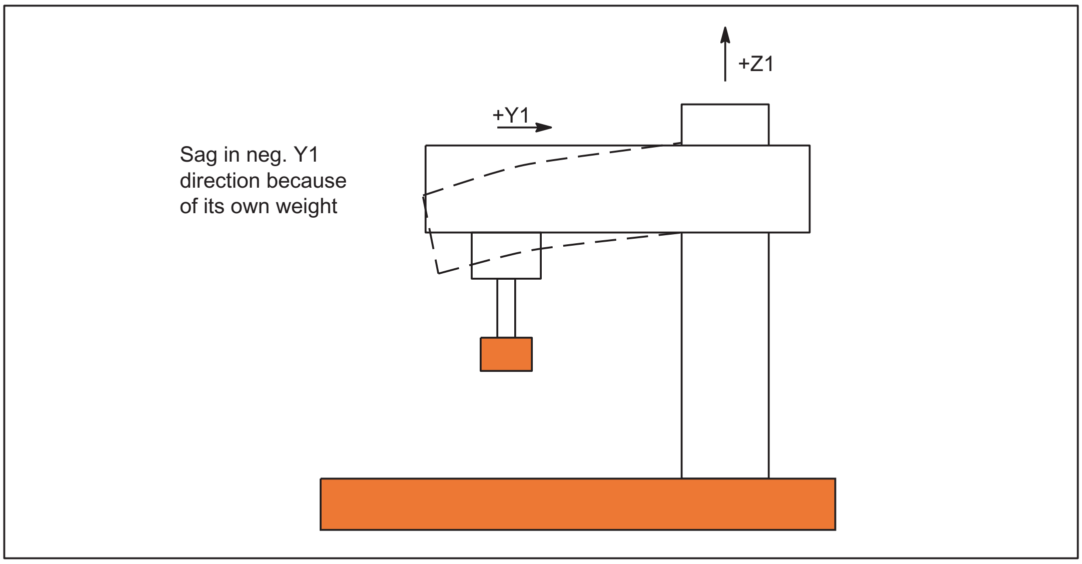


Figure 4-5 Example of sag caused by own weight

The error must be recorded in the form of a compensation table that contains a compensation value for the Z1 axis for every actual value position in the Y1 axis. It is sufficient to enter the compensation values for the interpolation points.

When the Y1 axis traverses, the control calculates the corresponding compensation value in the Z1 axis in interpolation cycles performing linear interpolation for positions between the interpolation points. This compensation is sent to the position control loop as an additional setpoint. A positive compensation value causes the corresponding machine axis to move in the negative direction.

Depending on the requirement, several compensation relations can be defined for one axis. The total compensation value results from the sum of all the compensation values of this axis.

Setting options

The many ways in which the compensation value for sag compensation can be produced/influenced are listed below (see diagram below).

1. An axis can be defined as the input variable (base axis) for **several** compensation tables (settable via system variables).
2. An axis can be defined as the recipient of the output variable (compensation axis) of **several** compensation tables (settable via system variable). The total compensation value is derived from the sum of the individual compensation values.

   The following definitions apply for the maximum number of possible compensation tables:

   – Maximum number of tables available for all axes:

     2 * maximum number of axes of system

   – Maximum number of tables configured for one particular compensation axis:

     1 * maximum number of axes of system
3. An axis can be both a base axis and a compensation axis at any one time. The programmed (required) position setpoint is always used to calculate the compensation values.
4. The scope of action of the compensation (starting and end position of the base axis) and the distance between the interpolation points can be defined for every compensation table (settable via system variables).
5. Compensation can be direction-dependent (settable via system variables).
6. Every compensation table has a modulo function for cyclic evaluation (settable via system variables).
7. A weighting factor by which the table value is multiplied (definable as a setting data which can therefore be altered by the part program, PLC or the user at any time) can be introduced for every compensation table.

8. Every compensation table can be multiplied with any other compensation table in pairs (i.e. also with itself) using the "table multiplication" function. A system variable is used to link the multiplication. The product is added to the total compensation value of the compensation axis.
9. The following options are available to activate the compensation:
   - Machine data:

     MD32710 $MA_CEC_ENABLE[<AXi>] (enable sag compensation)

     enables the sum of all compensation relationships for machine axis <AXi>.
   - With the setting data:

     SD41300 $SN_CEC_TABLE_ENABLE[<t>] (pre-assignment for the compensation table)

     evaluation of the compensation table [<t>] is enabled.

     It is thus possible e.g. to alter the compensation relationships either from the part program or from the PLC user program (e.g. switching over the tables), depending on the machining requirements.

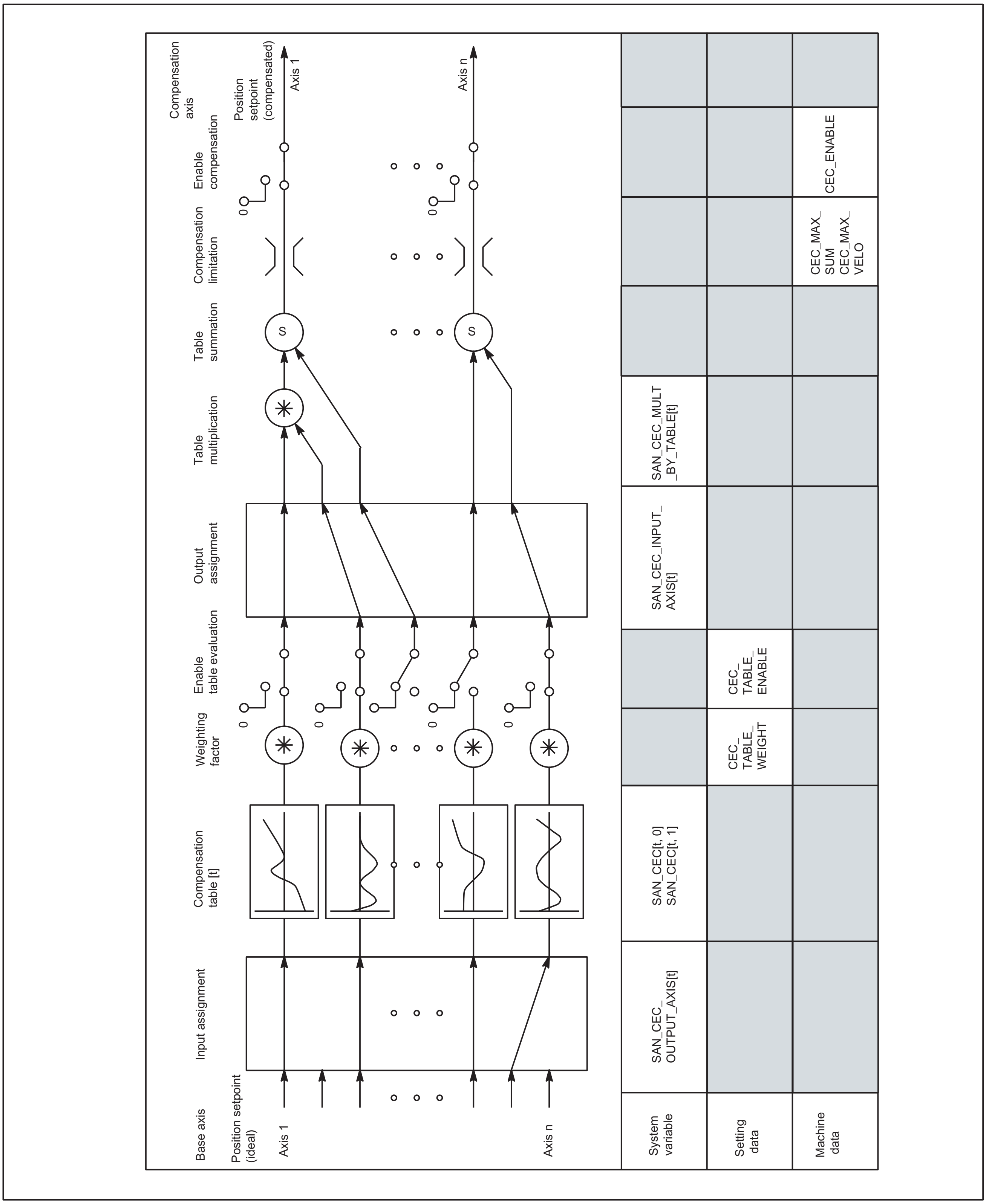


Figure 4-6 Generation of compensation value for sag compensation

Complex compensation

Since it is possible to use the position of an axis as the input quantity (base axis) for several tables, to derive the total compensation value of an axis from several compensation relationships (tables) and to multiply tables, it is also possible to implement sophisticated and complex beam sag and angularity error compensation systems.

This function also makes it possible to deal with different error sources efficiently. For example, it is possible to combine a table with a modulo function for a periodic recurring error component with a second table without a modulo function for an aperiodic error component for the same axis.

Leadscrew errors can also be compensated with this function by parameterizing an identical axis for the base and compensation axes. However, in contrast to the MSEC, measuring-system switchovers are not automatically registered in this case.

Preconditions / activation

The "sag compensation" function does not become active until the following conditions are fulfilled:

- The function has been activated for the relevant machine axis (compensation axis):

  MD32710 $MA_CEC_ENABLE[<AXi>] = 1
- The compensation values are stored in the static user memory and are active (after POWER ON).
- Evaluation of the relevant compensation table has been enabled:

  SD41300 $SN_CEC_TABLE _ENABLE[<t>] = 1
- The current measuring system of the base and compensation axes has been referenced:

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

As soon as these conditions have been fulfilled the setpoint position of the compensation axis is altered in all modes with reference to the setpoint position of the base axis and the corresponding compensation value and is then immediately traversed by the machine axis.

If the reference is then lost, e.g. because the encoder frequency has been exceeded (DB31, ... DBX60.4 or 60.5 = 0), compensation processing is deactivated.

4.4.3.2 Commissioning

Number of compensation interpolation points

The number of required interpolation points in the compensation table must be defined for every compensation relationship and the memory space required is reserved using the following machine data:

MD18342 $MN_MM_CEC_MAX_POINTS[<index>] (maximum number of interpolation points for sag compensation), with index = 0, 1, 2, ... (2 * maximum number of axes) - 1

The required number of interpolation points of a compensation table is calculated as follows:

$$\text{MM\_CEC\_MAX\_POINTS [t]} = \frac{\text{\$AN\_CEC\_MAX [t]} - \text{\$AN\_CEC\_MIN [t]}}{\text{\$AN\_CEC\_STEP [t]}} + 1$$

Table parameters

The position-related compensation values as well as additional table parameters should be saved for every compensation relationship in the form of system variables:

- $AN_CEC[<index>,<N>] (compensation value for interpolation point <N> of compensation table [<t>])

  <N> = interpolation point (position of the basic axis), with 0 ≤ N ≤ (value of MD18342) - 1

  The particular compensation value of the compensation axis must be entered in the table for each individual interpolation point.
- $AN_CEC_INPUT_AXIS[<index>] (basic axis)

  Name of machine axis whose setpoint is to be used as the input for the compensation table [<t>].
- $AN_CEC_OUTPUT_AXIS[<index>] (compensation axis)

  Name of machine axis to which the output of the compensation table [<t>] is applied.

  Note

  In multi-channel systems the "general axis names" AX1 ... must be defined, if the names of machine axis and channel axis are identical.

- $AN_CEC_STEP[<index>] (interpolation point distance)

  The interpolation point distance defines the distance between the input values for the compensation table [<t>].
- $AN_CEC_MIN[<index>] (initial position)

  The initial position is the basic axis position at which the compensation table [<t>] begins (≙ interpolation point 0).

  The compensation value for the initial position is $AN_CEC [<t>,0].

  The compensation value of interpolation point 0 is used for all positions smaller than the initial position (exception: table with modulo functions).
- $AN_CEC_MAX[<index>] (end position)

  The end position is the basic axis position at which the compensation table ends, = interpolation point <k>.

  The compensation value for the end position is $AN_CEC [<index>,**<k>**].

  The compensation value of interpolation point <k> is used for all positions larger than the end position (exception: table with modulo functions).

  The following boundary conditions apply to interpolation point k:

  – For k = MD18342 - 1:

    The compensation table is fully utilized!

  – For k < MD18342 - 1:

    The compensation table is not fully utilized. The compensation values entered that are greater than k are ignored.

  – For k > MD18342 – 1:

    The compensation table is limited by a control function which reduces the end position. Compensation values that are greater than k are ignored.

- $AN_CEC_DIRECTION[<index>] (direction-dependent compensation)

  This system variable is used to set whether the compensation table [<t>] should apply to both traversing directions of the basic axis or only either the positive or negative direction:

  – $AN_CEC_DIRECTION[<index>] = 0:

    Table applies to both directions of travel of the basic axis

  – $AN_CEC_DIRECTION[<index>] = 1:

    Table applies only to the positive traversing direction of the basic axis

  – $AN_CEC_DIRECTION[<index>] = -1:

    Table applies only to the negative traversing direction of the basic axis

  Possible applications:

  Position-dependent backlash compensation can be implemented using two tables, one of which affects the positive traversing direction, the other of which affects the negative traversing direction of the same axis.

- $AN_CEC_MULT_BY_TABLE [<index>] (table multiplication)

  With the table multiplication function, the compensation values of any table can be multiplied by those of any other table (or even by the same table). The product is added as an additional compensation value to the total compensation value of the compensation table.

  Syntax:

  $AN_CEC_MULT_BY_TABLE[<index>] = <number>

  with:

  <index> = table index of table 1 of the compensation axis

  <number> = table number of table 2 of the compensation axis, with
  table number = table index + 1

- $AN_CEC_IS_MODULO[<index>] (compensation with modulo function)

  System variable to activate/deactivate the compensation with modulo function:

  – $AA_CEC_COMP_IS_MODULO[<index>] = **0**: Compensation **without** modulo function

  – $AA_CEC_COMP_IS_MODULO[<index>] = **1**: Compensation **with** modulo function

  When compensation with modulo function is activated, the compensation table is repeated cyclically, i.e. the compensation value at position $AN_CEC_MAX[<index>] corresponding to interpolation point $AN_CEC[<index>,<k>] is immediately followed by the compensation value at position $AN_CEC_MIN[<index>] corresponding to interpolation point $AN_CEC[<index>,0].

  These two compensation values should be the same as otherwise the compensation value jumps from MAX to MIN at the transition point and vice versa.

  If modulo compensation is to be implemented with a modulo rotary axis as basic axis, the compensation table used has to be modulo calculated as well.

Example:

MD30300 $MA_IS_ROT_AX[AX1]=1 ; rotary axis

MD30310 $MA_ROT_IS_MODULO[AX1]=1 ; modulo 360°

$AN_CEC_INPUT_AXIS[0] = AX1

$AN_CEC_MIN[0] = 0.0

$AN_CEC_MAX[0] = 360.0

$AN_CEC_IS_MODULO[0] = 1

System of units

Table parameters containing position information are automatically converted when the system of units is changed (change from MD10240 $MN_SCALING_SYSTEM_IS_METRIC).

The position information is always interpreted in the current measuring system. Conversion must be implemented externally.

Automatic conversion of the position data can be configured as follows:

MD10260 $MN_CONVERT_SCALING_SYSTEM = 1

With this setting, the following axial machine data is activated:

MD32711 $MA_CEC_SCALING_SYSTEM_METRIC (measuring system for sag compensation)

The measuring system for all tables effective for this axis is set in this machine data. Hereby, all position entries are interpreted together with the calculated total compensation value in the configured measuring system. External conversions of position information are no longer necessary with a measuring system change.

Monitoring

To avoid excessive velocities and acceleration rates on the machine axis as a result of applying sag compensation, the total compensation value is monitored and limited to a maximum value.

The maximum possible total compensation value for sag compensation is defined on an axis-for-axis basis using the machine data:

MD32720 $MA_CEC_MAX_SUM (maximum compensation value for sag compensation)

If the determined total compensation value is greater than the maximum value, then a corresponding alarm is output. Program processing is not interrupted. The compensation value output as an additional setpoint is limited to the maximum value.

Further, changing the total compensation value is also axially limited:

MD32730 $MA_CEC_MAX_VELO (velocity change for sag compensation)

The specified value acts as a factor and is referred to the maximum axis velocity (MD32000 $MA_MAX_AX_VELO).

An appropriate alarm is signaled when the limit value is exceeded. Program processing is not interrupted. The path not covered because of the limitation is made up as soon as the compensation value is no longer subject to limitation.

4.4.3.3 Examples

Compensation table for sag compensation of the Y1 axis

The following example shows the compensation table for sag compensation of the Y1 axis. Depending on the position of the Y1 axis, a compensation value is applied to the Z1 axis. The 1st compensation table with index = 0 is used for this.

Program code	**Comment**
%_N_NC_CEC_INI	;
CHANDATA(1)	;
$AN_CEC[0,0]=0	;1st compensation value (interpolation point 0)
	; for Z1: ±0µm
$AN_CEC[0,1]=0.01	;2nd compensation value (interpolation point 1)
	; for Z1: +10µm
$AN_CEC[0,2]=0.012	;3rd compensation value (interpolation point 2)
	; for Z1: +12µm
...	;
$AN_CEC[0,100]=0	; Last compensation value (interpolation point 101)
	; for Z1: ±0µm
$AN_CEC_INPUT_AXIS[0]=(AX2)	; basic axis Y1
$AN_CEC_OUTPUT_AXIS[0]=(AX3)	; compensation axis Z1
$AN_CEC_STEP[0]=8	; distance between interpolation points 8.0 mm
$AN_CEC_MIN[0]=-400.0	; Compensation starts at Y1=-400 mm
$AN_CEC_MAX[0]=400.0	; Compensation ends at Y1=+400 mm
$AN_CEC_DIRECTION[0]=0	; Table applies for both directions of travel of Y1.
$AN_CEC_MULT_BY_TABLE[0]=0	;
$AN_CEC_IS_MODULO[0]=0	;compensation without modulo function
M17	;

For this example, the configured number of interpolation points must be ≥ 101:

MD18342 $MN_MM_CEC_MAX_POINTS[0] ≥ 101

The memory required in the static user memory is at least 808 bytes (8 bytes per compensation value).

Application for table multiplication

The following example for the compensation of machine foundation sagging illustrates an application of table multiplication.

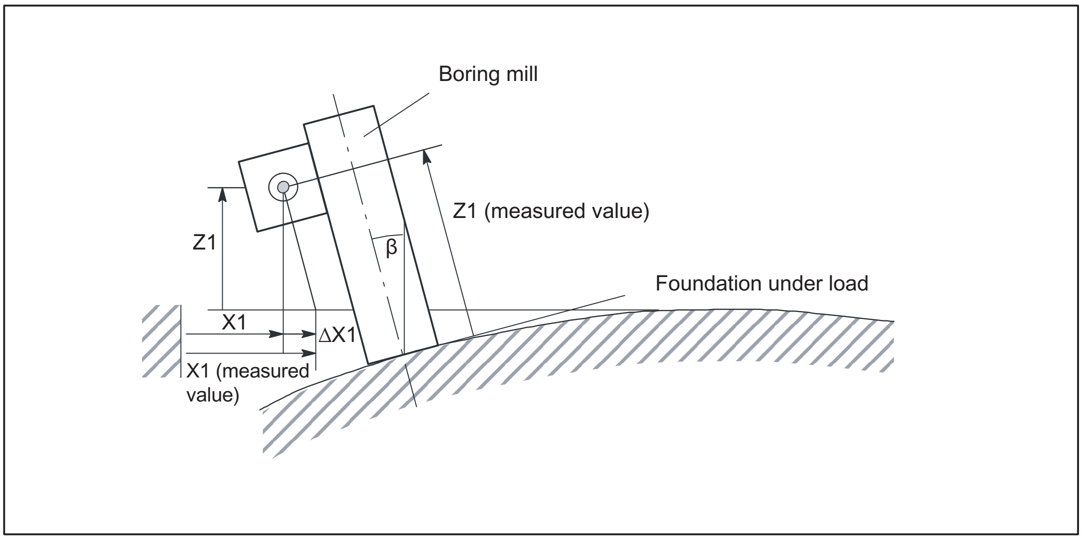


Figure 4-7 Compensation of sag in a foundation

On large machines, sagging of the foundation can cause inclination of the whole machine. For the boring mill shown above as an example, the compensation of the X1 axis depends on the position of the X1 axis itself, since this determines the angle of inclination β, and on the height Z1 of the drill.

The compensation values of the X1 and Z1 axes should be linked as follows for a compensation:

$\Delta X1 = Z1 * \sin\beta(X1) \approx Z1 * \beta(X1)$

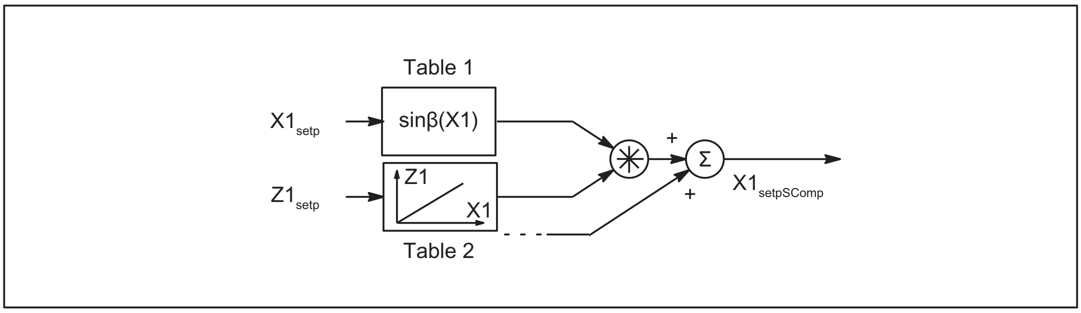


Figure 4-8 Table multiplication

Compensation table 1 (index = 0) describes the reaction of axis X1 on axis X1 (sine of the position-dependent tilting angle β(X1)).

Compensation table 2 (index = 1) describes the reaction of axis Z1 on axis X1 (linear).

In table 1, the multiplication of table 1 (index = 0) with table 2 is to be selected:

$AN_CEC_MULT_BY_TABLE[0] = 2

Input of compensation values in a grid structure

The compensation values of the Z axis sag on flat bed machines are often measured in practice at various points as a function of the X and Y coordinates. Under these preconditions, it is useful to enter the measured compensation values according to a grid-type distribution. The interpolation points with the relevant compensation values are positioned at the intersections of the grid (X-Y plane). Compensation values between these interpolation points are interpolated linearly by the control.

The following example explains in more detail how sag and angularity compensation can be implemented by a grid of 4 x 5 (lines x columns) in size. The size of the whole grid is 2000 x 900 mm^2. The compensation values are each measured in steps of 500 mm along the X axis and 300 mm along the Y axis.

Note

The following interdependencies apply for the maximum dimension of the grid (number of lines and columns):

- The number of lines depends on the number of axes in the system (dependent on the NCU type).
- The number of columns is dependent on the maximum number of values which can be entered in a compensation table (up to a total of 2000 values).

The number of lines and columns is set via the following machine data:

MD18342 $_MN_MM_CEC_MAX_POINTS (maximum number of interpolation points for sag compensation)

⚠ CAUTION

Memory-configuring machine data

Settings made to the MD18342 machine data cause the non-volatile NC user memory to be automatically re-allocated on system power-on. All the user data in the NC user memory (e.g. drive and HMI machine data, tool offsets, part programs, etc.) is deleted.

Save the user data before setting the machine data.

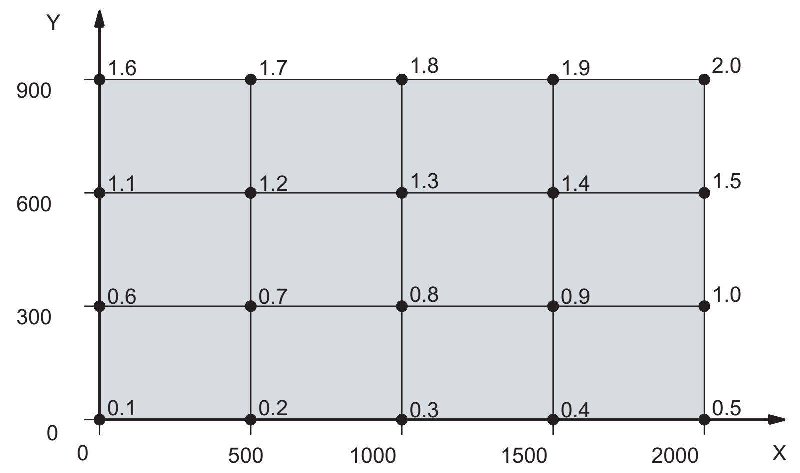


Figure 4-9 Compensation values of z axis with chessboard-like distribution of x-y plane

The application example can be realized with the following part program code:

```
$MA_CEC_ENABLE[Z1]= FALSE          ; Deactivate compensation
                                   ; by setting to FALSE.
                                   ; The table values can then be
                                   ; altered without generation of
                                   ; alarm 17070.
NEWCONF                            ; Activate $MA_CEC_ENABLE
;Define values f_i(x) in the f tables:
;Function values f_1(x) for table with index [0]
$AN_CEC[0,0]=0.1
$AN_CEC[0,1]=0.2
$AN_CEC[0,2]=0.3
$AN_CEC[0,3]=0.4
$AN_CEC[0,4]=0.5

;Function values f_2(x) for table with index [1]
$AN_CEC[1,0]=0.6
$AN_CEC[1,1]=0.7
$AN_CEC[1,2]=0.8
$AN_CEC[1,3]=0.9
$AN_CEC[1,4]=1.0
```

```
;Function values f_3(x) for table with index [2]
$AN_CEC[2,0]=1.1
$AN_CEC[2,1]=1.2
$AN_CEC[2,2]=1.3
$AN_CEC[2,3]=1.4
$AN_CEC[2,4]=1.5

;Function values f_4(x) for table with index [3]
$AN_CEC[3,0]=1.6
$AN_CEC[3,1]=1.7
$AN_CEC[3,2]=1.8
$AN_CEC[3,3]=1.9
$AN_CEC[3,4]=2.0

;Enable evaluation of f tables with compensation values
$SN_CEC_TABLE_ENABLE[0]=TRUE
$SN_CEC_TABLE_ENABLE[1]=TRUE
$SN_CEC_TABLE_ENABLE[2]=TRUE
$SN_CEC_TABLE_ENABLE[3]=TRUE

;Define weighting factor of f tables
$SN_CEC_TABLE_WEIGHT[0]=1.0
$SN_CEC_TABLE_WEIGHT[1]=1.0
$SN_CEC_TABLE_WEIGHT[2]=1.0
$SN_CEC_TABLE_WEIGHT[3]=1.0

;Changes to the following table parameters do not take effect until
;a Power On
;Define basic axis X1
$AN_CEC_INPUT_AXIS[0]=(X1)
$AN_CEC_INPUT_AXIS[1]=(X1)
$AN_CEC_INPUT_AXIS[2]=(X1)
$AN_CEC_INPUT_AXIS[3]=(X1)

;Define compensation axis Z1
$AN_CEC_OUTPUT_AXIS[0]=(Z1)
$AN_CEC_OUTPUT_AXIS[1]=(Z1)
$AN_CEC_OUTPUT_AXIS[2]=(Z1)
$AN_CEC_OUTPUT_AXIS[3]=(Z1)

;Define distance between interpolation points for compensation values in f tables
$AN_CEC_STEP[0]=500.0
$AN_CEC_STEP[1]=500.0
$AN_CEC_STEP[2]=500.0
$AN_CEC_STEP[3]=500.0
```

```
;Compensation starts at X1=0
$AN_CEC_MIN[0]=0.0
$AN_CEC_MIN[1]=0.0
$AN_CEC_MIN[2]=0.0
$AN_CEC_MIN[3]=0.0

;Compensation ends at X1=2000
$AN_CEC_MAX[0]=2000.0
$AN_CEC_MAX[1]=2000.0
$AN_CEC_MAX[2]=2000.0
$AN_CEC_MAX[3]=2000.0
;Values of f tables with index [t1] are multiplied by values in g tables
;by the number [t2]
;in accordance with the rule of calculation specified above
$AN_CEC_MULT_BY_TABLE[0] = 5
$AN_CEC_MULT_BY_TABLE[1] = 6
$AN_CEC_MULT_BY_TABLE[2] = 7
$AN_CEC_MULT_BY_TABLE[3] = 8

;Define the g table values for g_i(y) :
;Function values g_1(x) for table with index [4]
$AN_CEC[4,0]=1.0
$AN_CEC[4,1]=0.0
$AN_CEC[4,2]=0.0
$AN_CEC[4,3]=0.0

;Function values g_2(x) for table with index [5]
$AN_CEC[5,0]=0.0
$AN_CEC[5,1]=1.0
$AN_CEC[5,2]=0.0
$AN_CEC[5,3]=0.0

;Function values g_3(x) for table with index [6]
$AN_CEC[6,0]=0.0
$AN_CEC[6,1]=0.0
$AN_CEC[6,2]=1.0
$AN_CEC[6,3]=0.0

;Function values g_4(x) for table with index [7]
$AN_CEC[7,0]=0.0
$AN_CEC[7,1]=0.0
$AN_CEC[7,2]=0.0
$AN_CEC[7,3]=1.0
```

```
;Enable evaluation of g tables with compensation values
$SN_CEC_TABLE_ENABLE[4]=TRUE
$SN_CEC_TABLE_ENABLE[5]=TRUE
$SN_CEC_TABLE_ENABLE[6]=TRUE
$SN_CEC_TABLE_ENABLE[7]=TRUE

;Define weighting factor for g tables
$SN_CEC_TABLE_WEIGHT[4]=1.0
$SN_CEC_TABLE_WEIGHT[5]=1.0
$SN_CEC_TABLE_WEIGHT[6]=1.0
$SN_CEC_TABLE_WEIGHT[7]=1.0

;Changes to the following table parameters do not take effect until
;a Power On
;Define basic axis Y1
$AN_CEC_INPUT_AXIS[4]=(Y1)
$AN_CEC_INPUT_AXIS[5]=(Y1)
$AN_CEC_INPUT_AXIS[6]=(Y1)
$AN_CEC_INPUT_AXIS[7]=(Y1)

;Define compensation axis Z1
$AN_CEC_OUTPUT_AXIS[4]=(Z1)
$AN_CEC_OUTPUT_AXIS[5]=(Z1)
$AN_CEC_OUTPUT_AXIS[6]=(Z1)
$AN_CEC_OUTPUT_AXIS[7]=(Z1)
;Define distance between interpolation points for compensation values in g tables
$AN_CEC_STEP[4]=300.0
$AN_CEC_STEP[5]=300.0
$AN_CEC_STEP[6]=300.0
$AN_CEC_STEP[7]=300.0

;Compensation starts at Y1=0
$AN_CEC_MIN[4]=0.0
$AN_CEC_MIN[5]=0.0
$AN_CEC_MIN[6]=0.0
$AN_CEC_MIN[7]=0.0

;Compensation ends at Y1=900
$AN_CEC_MAX[4]=900.0
$AN_CEC_MAX[5]=900.0
$AN_CEC_MAX[6]=900.0
$AN_CEC_MAX[7]=900.0
$MA_CEC_ENABLE[Z1]=TRUE           ;Activate compensation again
NEWCONF
```

```
;Carry out a program test to check whether the
;compensation is effective
G01 F1000 X0 X0 Z0 G90
R1=0 R2=0
LOOP_Y:
LOOP_X:
STOPRE
X=R1 Y=R2
M0                                  ; Wait to check the CEC value
R1=R1+500
IF R1 <=2000 GOTOB LOOP_X
R1=0
R2=R2+300
IF R2<=900 GOTOB LOOP_Y
```

Note

You can read the compensation value under variable "Sag + temperature compensation" on the user interface. To do so, select softkey "Diagnosis" followed by softkey "Service axis". The currently effective compensation value is displayed next to the "Sag + temperature compensation" variable.

```
;to prepare the table configuration, the Power On
;machine data is set
;cec.md:
;Set option data for commissioning
;Define the number of interpolation points in the compensation tables
;Machine data configures the memory
$MN_MM_CEC_MAX_POINTS[0]=5
$MN_MM_CEC_MAX_POINTS[1]=5
$MN_MM_CEC_MAX_POINTS[2]=5
$MN_MM_CEC_MAX_POINTS[3]=5
$MN_MM_CEC_MAX_POINTS[4]=4
$MN_MM_CEC_MAX_POINTS[5]=4
$MN_MM_CEC_MAX_POINTS[6]=4
$MN_MM_CEC_MAX_POINTS[7]=4
$MA_CEC_MAX_SUM[AX3]=10.0           ; Define the maximum
                                    ; total compensation value
$MA_CEC_MAX_VELO[AX3]=100.0         ; Limit the maximum changes in the
                                    ; total compensation value
M17
```

Explanation

The compensation values cannot be entered directly as a 2-dimensional grid. Compensation tables in which the compensation values are entered must be created first.

A compensation table contains the compensation values of one line (four lines in the example, i.e. four compensation tables). The compensation values 0.1 to 0.5 are entered in the first table in the example, the compensation values 0.6 to 1.0 in the second table, and so on. The compensation tables are referred to below as f tables and their values as f_i(x) (i=number of table).

The compensation values of f tables are evaluated by multiplying them by other tables. The latter are referred to below as g tables and their values as g_i(y). The number of f tables and g tables is equal (four in the example).

In g tables, one compensation value in each table is set to 1 and all the others to 0. The position of compensation value 1 within the table is determined by the table number. In the first g table, compensation value 1 is positioned at the first interpolation point and, in the second g table, at the second interpolation point, etc. By multiplying g tables by f tables, the correct compensation value in each f table is selected by multiplying it by 1. All irrelevant compensation values are concealed through multiplication by 0.

Using this scheme, compensation value D_z at position (x/y) is calculated according to the following equation:

D_z(x/y)=f_1(x)*g_1(y) + f_2(x)*g_2(y) + ...

When the compensation value for the actual position of the machine spindle is calculated, the f table values are multiplied by the g table values according to this rule.

Applied to the example, this means, for instance that compensation value D_z(500/300) is calculated by multiplying each of the function values f_i(500) in the f tables by the function values g_i(300) in the g tables:

D_z(500/300) = f_1(1000)*g_1(300) + f_2(1000)*g_2(300) + f_3(1000)*g_3(300) + f_4(1000)*g_4(300)

D_z(500/300) = 0.2*0 + 0.7*1 + 1.2*0 + 1.7*0 = 0.7

4.4.4 Direction-dependent leadscrew error compensation

4.4.4.1 Description of functions

If the direction-dependent differences at the compensation points are excessively high, for an inconsistent backlash or for extremely high demands placed on the precision, then it may be necessary to apply direction-dependent compensation of the leadscrew error or measuring system error (for direct position sensing).

Direction-dependent leadscrew error compensation

For the "direction-dependent leadscrew error compensation" ("direction-dependent LEC" or also "Bidirectional LEC") , two compensation tables are used for each axis. One compensation table for the positive and one compensation table for the negative traversing direction. The deviation at the particular compensation point is entered as difference between the ideal setpoint and measured actual value in the compensation tables. The control automatically calculates compensation values of intermediate values using linear interpolation.

Preconditions / activation

The "direction-dependent LEC" is implemented in the SINUMERIK control as a special case of "sag compensation". This is the reason that the preconditions and conditions of "sag compensation" apply (see "Compensation of sag and angularity errors (Page 243)").

The activation of the compensation can be checked using a reference measurement, e.g. using the laser interferometer or in the simplest case, using the service display of the particular axis.

Note

If the "direction-dependent LEC" is used in parallel to the sag compensation and compensation of the angularity, then the secondary conditions of these functions must be taken into consideration together, e.g. the assignment of tables <t> to the particular function.

4.4.4.2 Commissioning

Measuring the error or compensation values

When commissioning the "direction-dependent LEC" - just the same as when commissioning the "direction-dependent LEC" - direction-dependent error curves for each axis are determined using a suitable measuring device (e.g. laser interferometer) (see Section "Compensation of leadscrew errors and measuring system errors (Page 238)"). A part program with measurement points and wait times should be generated in order to perform the measurement (see Section "Example (Page 264)"): Program "BI_SSFK_MESS_AX1_X.MPF").

Because the various measuring devices offer different support options for the practical implementation in conjunction with a SINUMERIK control, this process is only generally described in the following referred to a control.

Note

The measurement for determining the leadscrew error should only be carried out during the first commissioning if, in the machine data, the traversing directions of the axes in relation to the machine coordinate system have been correctly set.

Carrying out commissioning

1. Specify the number of compensation interpolation points (also see Section "Compensation for droop and angularity error: Commissioning (Page 247)")

   Each axis should be assigned to one compensation table each for the positive and negative traversing directions. The number of compensation interpolation points is defined using the following machine data:

   MD18342 $MN_MM_CEC_MAX_POINTS[<t>] (maximum number of interpolation points for sag compensation)

   with: <t> = Index of compensation table

   Permissible range: 0 ≤ t < 2 * maximum number of axes

⚠ CAUTION
User data loss ALARM 4400 is output when changing MD18342: "Reorganization of the buffered memory!" In order that an automatic memory configuration is possible but keeping all of the data that has been entered up until now, **no** system boot (POWER ON) must be executed without first performing a series machine startup.

   Example:

   MD18342 [0] = 11; 11 interpolation points for the 1st table, e.g. positive traversing direction, X axis

   MD18342 [1] = 11; 11 interpolation points for the 2nd table, e.g. negative traversing direction, X axis

   MD18342 [2] = 21; 21 interpolation points for the 3rd table, e.g. positive traversing direction, Y axis

   MD18342 [3] = 21; 21 interpolation points for the 4th table, e.g. positive traversing direction, Y axis

   …

   MD18342 [61] = ...; number of interpolation points for the 62nd table

2. Perform the series machine startup:

   – Generate an NC archive with the entries in MD18342 [<t>].

   – Read-in the generated NC archive.

     Note: The NC memory is configured as a result.

   The compensation tables are now available.

3. Generate the tables with compensation values for the particular axes and traversing directions as part program (see Section "Example (Page 264)": Program "BI_SSFK_MESS_AX1_X.MPF").

4. Execute the part program with compensation values in the control.

   AUTOMATIC mode > select program > NC start

   Note

   Each time before reading-in the compensation tables, the following parameters should always be set to **0** and then to activate, always be set to **1**:

   MD32710 $MA_CEC_ENABLE[<AXi>] (enable sag compensation) = **0** → **1**

   SD41300 $SN_CEC_TABLE _ENABLE[<t>] (enable the compensation table) = **0** → **1**

   The backlash should always be set to **0**:

   MD32450 $MA_BACKLASH [<e>] (backlash) = **0**

   with: <e> = Position measuring system

   The use of the program template "BI_SSFK_TAB_AX1_X.MPF" (see Section "Example (Page 264)") automates these tasks. When manually entering machine data, the generally applicable "Activate MD" or "Reset" should be observed.

   ---

5. POWER ON (warm restart).
6. Now, comparative measurements can be made using the laser interferometer.
7. To further improve the compensation results, it is also conceivable to correct individual compensation values in the program. A POWER ON is no longer necessary when reading-in the table again.

Note

Sequence for SINUMERIK 828D

For SINUMERIK 828D, steps **2** and **3** are eliminated. This is because when the "sag compensation, multi-dimensional" option is enabled, 8 tables each with 200 interpolation points per table for the compensation immediately become available. This cannot be extended!

Note

As described in step **5**, the compensation table is downloaded into the program memory as an executable program and is then transferred into the previously configured memory area of the control using an NC start. This procedure can be repeated for each table to ensure transparency. However, it is also possible to download all tables in an initialization step. The compensation values become active after MD32710[<AXi>] = 1 and a mandatory power POWER ON.

Note

NC_CEC.INI

The "NC_CEC.INI" file copied via "Commissioning" > "System data" (from the folder "NC active data" > "sag angularity comp") includes all negotiated sag/angularity and direction-dependent LEC tables.

Table parameters

The position-related compensations for the particular direction as well as additional table parameters in the form of system variables should be saved in the compensation table:

- $AN_CEC[<t>,<N>] (compensation value for interpolation point <N> of compensation table [<t>])
- $AN_CEC_INPUT_AXIS[<t>] (basic axis)
- $AN_CEC_OUTPUT_AXIS[<t>] (compensation axis)

> **Note**
>
> For the "direction-dependent LEC", the basis and compensation axis are identical.

- $AN_CEC_STEP[<t>] (interpolation point distance)
- $AN_CEC_MIN[<t>] (initial position)
- $AN_CEC_MAX[<t>] (end position)
- $AN_CEC_DIRECTION[<t>] (direction-dependent compensation)

  This system variable is used to set whether the compensation table [<t>] should apply to both positive and negative traversing directions of the basic axis:

  – $AN_CEC_DIRECTION[<t>] = 1:

    Table applies only to the positive traversing direction of the basic axis

  – $AN_CEC_DIRECTION[<t>] = -1:

    Table applies only to the negative traversing direction of the basic axis

    > **Note**
    >
    > The setting $AN_CEC_DIRECTION[<t>] = 0 (table is effective for both traversing directions of the basic axis) is **not** relevant for the "direction-dependent LEC".

- $AN_CEC_IS_MODULO[<t>] (compensation with modulo function)

> **Note**
>
> For a detailed description of these system variables (see Section "Compensation for droop and angularity error: Commissioning (Page 247)").

System of units

See Section "Compensation for droop and angularity error: Commissioning (Page 247)".

Monitoring

See Section "Compensation for droop and angularity error: Commissioning (Page 247)".

4.4.4.3 Example

The direction-dependent compensation tables of the X axis are shown in detail for a three-axis machine in the fallowing example:

Configuring

Number of compensation interpolation points:

MD18342 $MN_MM_CEC_MAX_POINTS[0] = 11 (Table 1: Axis X, **positive** traversing direction)

MD18342 $MN_MM_CEC_MAX_POINTS[1] = 11 (Table 2: Axis X, **negative** traversing direction)

Note

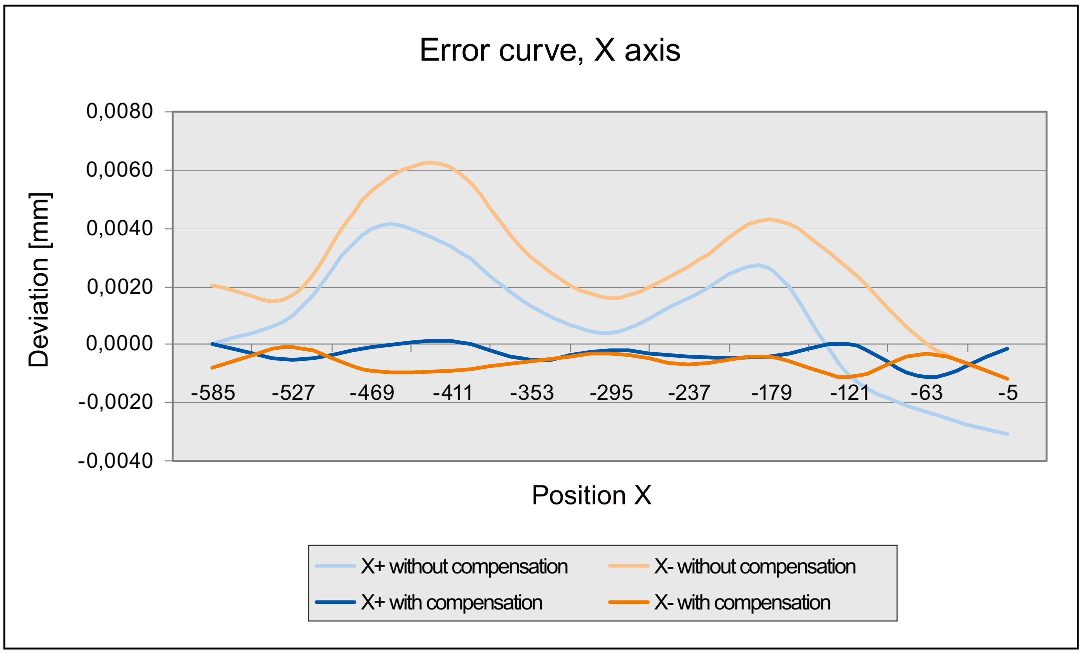
It is not necessary to define the number of interpolation points for SINUMERIK 828D, because when enabling the "sag compensation, multi-dimensional" option, immediately eight tables each with 200 interpolation points per table for the compensation are available. This cannot be extended!

Interpolation points

Table <t>	[0,<N>]										
Number of interpolation points	MD18342 $MN_MM_CEC_MAX_POINTS[0] = 11										
Interpolation point <N>	0	1	2	3	4	5	6	7	8	9	10
Position X	-585	-527	-469	-411	-353	-295	-237	-179	-121	-63	-5

Measurement

			Setpoint position	Deviation		Checking measurement	
	Position	Comp. No.	Measurement position [mm]	Direction + [mm]	Direction - [mm]	Direction + [mm]	Direction - [mm]
$AC_CEC_MIN[<t>]	-585	0	-585	0.0000	0.0020	0.0000	-0.0008
		1	-527	0.0010	0.0017	-0.0005	-0.0001
		2	-469	0.0040	0.0053	-0.0001	-0.0009
		3	-411	0.0034	0.0061	0.0001	-0.0009
		4	-353	0.0013	0.0030	-0.0005	-0.0006
		5	-295	0.0004	0.0016	-0.0002	-0.0003
		6	-237	0.0016	0.0027	-0.0004	-0.0007
		7	-179	0.0026	0.0043	-0.0004	-0.0004
		8	-121	-0.0010	0.0026	0.0000	-0.0011
		9	-63	-0.0023	0.0000	-0.0011	-0.0003
$AC_CEC_MAX[<t>]	-5	10	-5	-0.0031	-0.0012	-0.0001	-0.0012



Programming

The following program "BI_SSFK_TAB_AX1_X.MPF" includes the value assignments for the parameters of the two compensation tables (positive and negative traversing direction) of the X axis:

```
;direction-dependent LEC
;1st axis MX1
;Table 1 - positive traversing direction
;Table 2 - negative traversing direction
;-------------------------------------------------------------------------------------
CHANDATA(1)
$MA_CEC_ENABLE[AX1]=0                ;compensation OFF
$SN_CEC_TABLE_ENABLE[0]=0            ;lock Table 1
$SN_CEC_TABLE_ENABLE[1]=0            ;lock Table 2
NEWCONF
;-------------------------------------------------------------------------------------
$AN_CEC[0,0]=0                       ;1st compensation value (interpolation point 0)
$AN_CEC[0,1]=0.001                   ;2nd compensation value (interpolation point 1)
$AN_CEC[0,2]=0.004                   ;3rd compensation value (interpolation point 2)
$AN_CEC[0,3]=0.0034                  ;4th compensation value (interpolation point 3)
$AN_CEC[0,4]=0.0013                  ;5th compensation value (interpolation point 4)
$AN_CEC[0,5]=0.0004                  ;6th compensation value (interpolation point 5)
$AN_CEC[0,6]=0.0016                  ;7th compensation value (interpolation point 6)
$AN_CEC[0,7]=0.0026                  ;8th compensation value (interpolation point 7)
$AN_CEC[0,8]=-0.001                  ;9th compensation value (interpolation point 8)
$AN_CEC[0,9]=-0.0023                 ;10th compensation value (interpolation point 9)
$AN_CEC[0,10]=-0.0031                ;last compensation value (interpolation point 10)
$AN_CEC_INPUT_AXIS[0]=(AX1)          ;basic axis
$AN_CEC_OUTPUT_AXIS[0]=(AX1)         ;compensation axis
$AN_CEC_STEP[0]=58.0                 ;interpolation point distance
$AN_CEC_MIN[0]=-585.0                ;compensation starts
$AN_CEC_MAX[0]=-5.0                  ;compensation ends
$AN_CEC_DIRECTION[0]=1               ;Table applies for positive traversing directions
$AN_CEC_MULT_BY_TABLE[0]=0           ;no multiplication (not relevant here)
$AN_CEC_IS_MODULO[0]=0               ;compensation without modulo function
;-------------------------------------------------------------------------------------
$AN_CEC[1,0]=0.002                   ;(interpolation point 0)
$AN_CEC[1,1]=0.0017                  ;(interpolation point 1)
$AN_CEC[1,2]=0.0053                  ;(interpolation point 2)
$AN_CEC[1,3]=0.0061                  ;(interpolation point 3)
$AN_CEC[1,4]=0.003                   ;(interpolation point 4)
$AN_CEC[1,5]=0.0016                  ;(interpolation point 5)
$AN_CEC[1,6]=0.0027                  ;(interpolation point 6)
$AN_CEC[1,7]=0.0043                  ;(interpolation point 7)
```

```
$AN_CEC[1,8]=0.0026                 ;(interpolation point 8)
$AN_CEC[1,9]=0.000                  ;(interpolation point 9)
$AN_CEC[1,10]=-0.0012               ;(interpolation point 10)
$AN_CEC_INPUT_AXIS[1]=(AX1)         ;basic axis
$AN_CEC_OUTPUT_AXIS[1]=(AX1)        ;compensation axis
$AN_CEC_STEP[1]=58.                 ;interpolation point distance
$AN_CEC_MIN[1]=-585.0               ;compensation starts
$AN_CEC_MAX[1]=-5.0                 ;compensation ends
$AN_CEC_DIRECTION[1]=-1             ;Table applies for negative traversing directions
$AN_CEC_MULT_BY_TABLE[1]=0          ;no multiplication (not relevant here)
$AN_CEC_IS_MODULO[1]=0              ;compensation without modulo function (only for rotary axes)
;---------------------------------------------------------------------------------------
$MA_CEC_ENABLE[AX1]=1               ;compensation ON
$SN_CEC_TABLE_ENABLE[0]=1           ;enable Table 1
$SN_CEC_TABLE_ENABLE[1]=1           ;enable Table 2
NEWCONF
M17
```

Additional tables can be set-up, e.g. for axes Y and Z:

MD18342 $MN_MM_CEC_MAX_POINTS[**2**] = 90 (Table **3**: Axis **Y**, positive traversing direction)

MD18342 $MN_MM_CEC_MAX_POINTS[**3**] = 90 (Table **4**: Axis **Y**, negative traversing direction)

MD18342 $MN_MM_CEC_MAX_POINTS[**4**] = 50 (Table **5**: Axis **Z**, positive traversing direction)

MD18342 $MN_MM_CEC_MAX_POINTS[**5**] = 50 (Table **6**: Axis **Z**, negative traversing direction)

4.4.5 Extension of the sag compensation with NCU link - only 840D sl

Application

If a system is operated with NCU link, any number of axes of the NCU link grouping can be compensated mutually. The two axes that are coupled via sag compensation must be interpolated on one NCU.

See also:

- NCU link (Page 79)
- Axis container (Page 100)

Function

The parameterization of the sag compensation function is done by setting system variables of the form: $AN_CEC ...

These system variables are normally set via a part program that processes the NCK in a certain channel. The channel axis name can be used in the variables $AN_CEC_OUTPUT_AXIS or $AN_CEC_INPUT_AXIS. This way, each axis of the channel can be addressed, even if it is in a different NCU.

A program in Channel 2 can couple Axis ZZ with Axis XX via the following setting (the setting is valid for the table with the number 0):

$AN_CEC_INPUT_AXIS[0] = (XX)

$AN_CEC_OUTPUT_AXIS[0] = (ZZ)

This way AX3 on NCU-1 is "coupled" with AX2 on NCU-2 (see configuration 1).

The following variants can be used to parameterize if the axes to be coupled are on two different channels:

- **Version 1: "Programming with channel axis name"**:

  Two different part programs TP1 and TP2 are created, they are then processed in different channels.

  Axis "ZZ" is coupled to "XR":

  View from the part program TP1 in Channel 1:

  ```
  $AN_CEC_INPUT_AXIS[0] = (XR)
  ```

  View from the part program TP2 in Channel 2:

  ```
  $AN_CEC_OUTPUT_AXIS[0] = (ZZ)
  ```

  Axis AX2 on NCU2 is coupled with Axis AX1 on NCU1 upon restart after TP1 is executed in Channel 1 and TP2 is executed in Channel 2.

- **Version 2: "Programming with machine axis name"**:

  One part program is created that runs in any convenient channel of NCU1 and specifies the machine axis names together with the NCU numbers.

  Axis "ZZ" is coupled to "XR":

  ```
  $AN_CEC_INPUT_NCU[0]=1

  $AN_CEC_INPUT_AXIS[0] = (AX1)

  $AN_CEC_OUTPUT_NCU[0]=2

  $AN_CEC_OUTPUT_AXIS[0] = (AX2)
  ```

  The NCK monitors whether the axes on the local NCU have actually been interpolated, i.e. there is a channel that can program these axes. The local NCU is always the NCU on which the part program runs.

The following axes are allowed for NCU1 as input or output axes in Configuration 1: NC1_AX1, NC1_AX3, NC1_AX4, NC1_AX5, NC2_AX2, NC2_AX6

The data backup from the NCK always delivers the compensation data from the "machine axis name" view.

Note

The NCU number is to be programmed before the axis name. A sag compensation between NC1_AX1 and NC1_AX2 is not possible.

Assignment of the axes

The assignment of the input and output axes is done via the following system variables:

$AN_CEC_INPUT_NCU	and	$AN_CEC_INPUT_AXIS
$AN_CEC_OUTPUT_NCU	and	$AN_CEC_OUTPUT_AXIS

The system variables become effective only after a restart.

Data backup is always undertaken with machine axis names.

Note

The sag compensation can couple the axis only on one NCU, which can also be traversed from this NCU either via the part program or via a synchronized action.

These variables are set optionally if the axes (input and output) are not available on the local NCU. If one uses a channel axis name while programming $AN_CEC_INPUT_AXIS and $AN_CEC_OUTPUT_AXIS, then the system variables $AN_CEC_INPUT_NCU and $AN_CEC_OUTPUT_NCU become irrelevant.

The control checks whether the two axes can be interpolated from this NCU, i.e. a program can traverse the axes on this NCU. The axes can be assigned to different channels. Two axes belonging to different NCUs can also be compensated. Otherwise the control rejects it with Alarm 17040.

Both axes of compensation must be interpolated on one NCU, i.e. there may be one or two part programs that traverse the input and output axes on an NCU.

Axis container

The axis container is a grouping of similar axes. An axis from the group can be assigned to a channel axis. The assignment is variable, so that the axis in the channel always gets a new axis from the group assigned to it in the course of time. Thus, the part program can be programmed with one axis and it can gradually move different axes.

Example:

Four spindles are arranged on a drum. Each spindle carries a tool of the turning machine and it rotates the drum by 90 degrees in each cycle. The tools are transported from one machining station to the next in this way. The channel of the machining station must program only one spindle, though a new spindle is always changed. This is an axis container rotation.

The sag compensation can be combined with the axis container if it is in the basic position, i.e. $AN_AXCTAS == 0. Otherwise the programming is rejected with Alarm 17040.

"YY" is to be coupled to "XX" (see Configuration 2):

1. Programming with channel axis names

   ```
   $AN_CEC_INPUT_AXIS[0] = (XX)
   $AN_CEC_OUTPUT_AXIS[0] = (YY)
   ```

2. Programming with machine axis names

   ```
   $AN_CEC_INPUT_NCU[0]=1          ; optional ...
   $AN_CEC_INPUT_AXIS[0] = (AX2)
   $AN_CEC_OUTPUT_NCU[0]=2
   $AN_CEC_OUTPUT_AXIS[0] = (AX2
   ```

This couples Axis AX2 of NCU1 with Axis 2 of NCU2.

NOTICE
Axis confusion
YY is coupled to XX with each container rotation, there is a different axis behind YY now: YY "AX5 of NCU-1" is in configuration 3.
Other real axes are coupled after the rotation in this way: In this example, AX-5 of NCU-1 is coupled to AX-2 of NCU-1.

As a rule:

The coupling is created between two axes from the LAI layer so that other axes participate in the coupling after each axis container rotation. A new table must be activated for each container rotation to undertake a coupling exactly between two real axes.

Configuration example

The following figures (Configuration 1, Configuration 2 and Configuration 3) illustrate the axis configurations of an NCU link that is assembled from two NCUs.

The two channels CHAN-1 and CHAN-2 of NCU-1 are displayed in Configuration 1. Here, the channel axis names that are defined via the machine data $MC_AXCONF_CHANAX_NAME_TAB are entered. The channel configuration of the second NCU is not displayed.

All the axes interpolated by this NCU are compiled in the "Logical NCK machine axis image" (LAI layer). The assignment between channel and MCS axis layer is done via $MC_AXCONF_MACHAX_USED.

The assignment between the "Logical NCK machine axis image" and the real axes is undertaken via the machine data $MN_AXCONF_LOGICMACHAX_TAB. If ones pursues the connecting line that starts with channel axis ZZ, one ends at Axis AX-2 on NCU-2, i.e. to traverse the 2nd axis of NCU 2, the following command must be programmed in the 2nd channel of NCU 1: `"N2040 POS[ZZ]=10 FA[ZZ]=1000"`

Configuration 2 and Configuration 3 extend the figure of Configuration 1 by one axis container (CT1) that is set with machine data $MN_AXCT_AXCONF_ASSIGN_TAB1. The axis container is an overlapping object, i.e. each axis container exists only once for the whole NCU cluster.

For NCU 1, the participants in the axis container are channel axes YR and YY; the two channel axes from NCU2 are not displayed. The container contains the real axes NC1_AX5, NC1_AX6, NC2_AX1 and NC2_AX2. Container YR connects with NC2_AX1 and YY connects with NC2_AX2 during the ramp up. In Configuration 3, the container has rotated, i.e. the connection structure has changed. YR is now connected to NC2_AX2 and YY is connected to NC1_AX5.

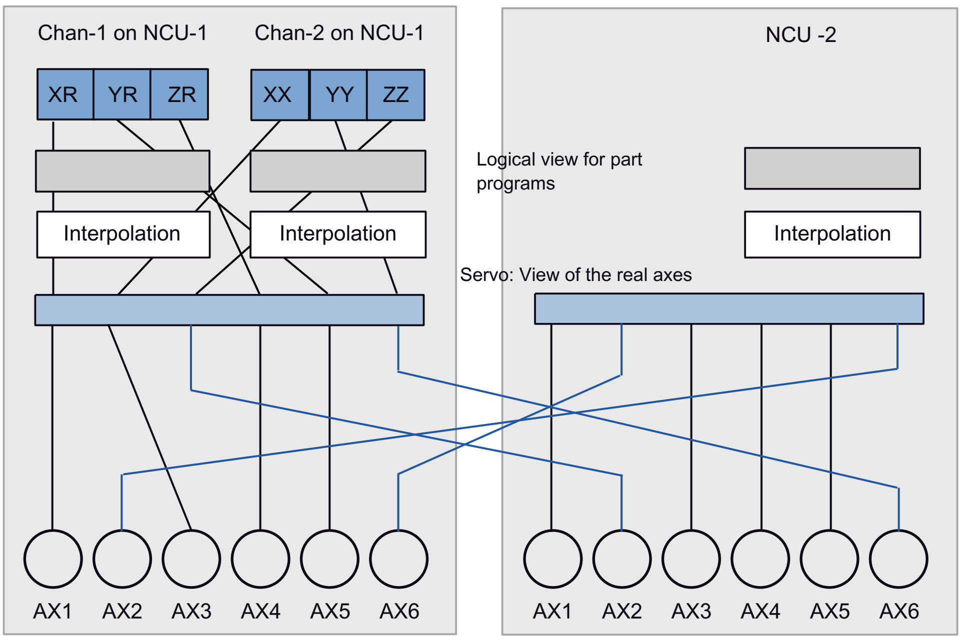


Figure 4-10 Configuration 1: NCU link from channel to real axis

Machine data of Configuration 1

```
; ########## NCU1 ##########
$MN_NCU_LINKNO = 1
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_LINK_NUM_OF_MODULES = 2
$MN_MM_SERVO_FIFO_SIZE = 3

$MN_ASSIGN_CHAN_TO_MODE_GROUP[1]=1

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "NC1_AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "NC1_AX3"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[3] = "NC1_AX4"
$MN_AXCONF_LOGIC_MACHAX_TAB[4] = "NC1_AX5"
$MN_AXCONF_LOGIC_MACHAX_TAB[5] = "NC2_AX6"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=5
$MC_AXCONF_MACHAX_USED[2]=4
$MC_AXCONF_MACHAX_USED[3]=0
$MC_AXCONF_MACHAX_USED[4]=0
$MC_AXCONF_MACHAX_USED[5]=0
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XR"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YR"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZR"

CHANDATA(2)
$MC_REFP_NC_START_LOCK=0
$MC_AXCONF_MACHAX_USED[0]=2
$MC_AXCONF_MACHAX_USED[1]=6
$MC_AXCONF_MACHAX_USED[2]=3
$MC_AXCONF_MACHAX_USED[3]=0
$MC_AXCONF_MACHAX_USED[4]=0
```

```
$MC_AXCONF_MACHAX_USED[5]=0
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YY"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZZ"
M30

; ########## NCU-2 ##########
$MN_NCU_LINKNO = 2
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_LINK_NUM_OF_MODULES = 2
$MN_MM_SERVO_FIFO_SIZE = 3
;
;
;
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "NC2_AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "NC1_AX6"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX3"
$MN_AXCONF_LOGIC_MACHAX_TAB[3] = "NC2_AX4"
$MN_AXCONF_LOGIC_MACHAX_TAB[4] = "NC2_AX5"
$MN_AXCONF_LOGIC_MACHAX_TAB[5] = "NC1_AX2"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=2
$MC_AXCONF_MACHAX_USED[2]=3
$MC_AXCONF_MACHAX_USED[3]=4
$MC_AXCONF_MACHAX_USED[4]=5
$MC_AXCONF_MACHAX_USED[5]=6
$MC_AXCONF_MACHAX_USED[6]=0
M30
```

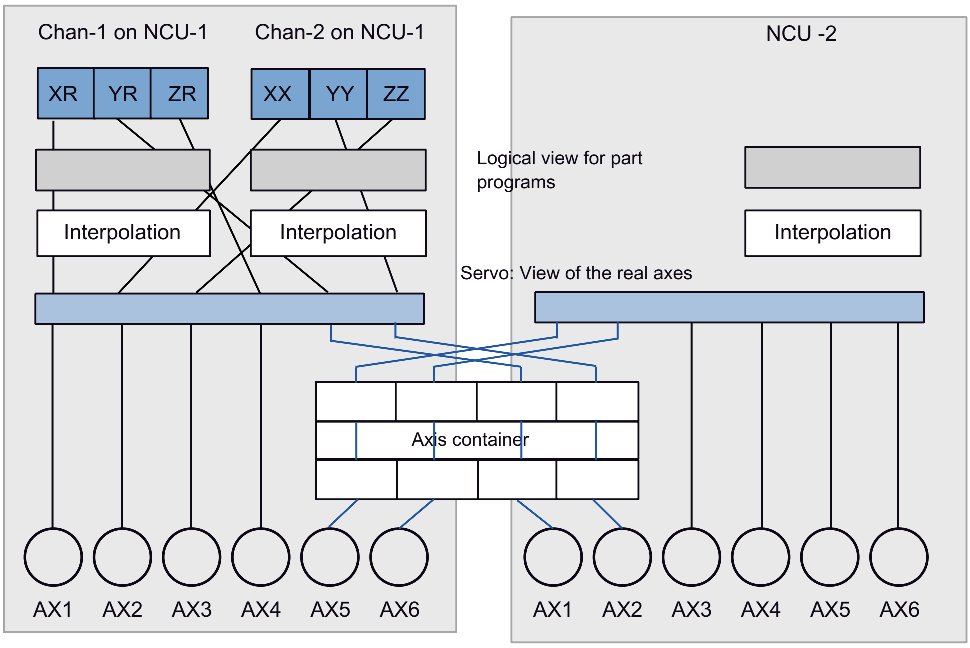


Figure 4-11 Configuration 2: NCU link with axis container in output state

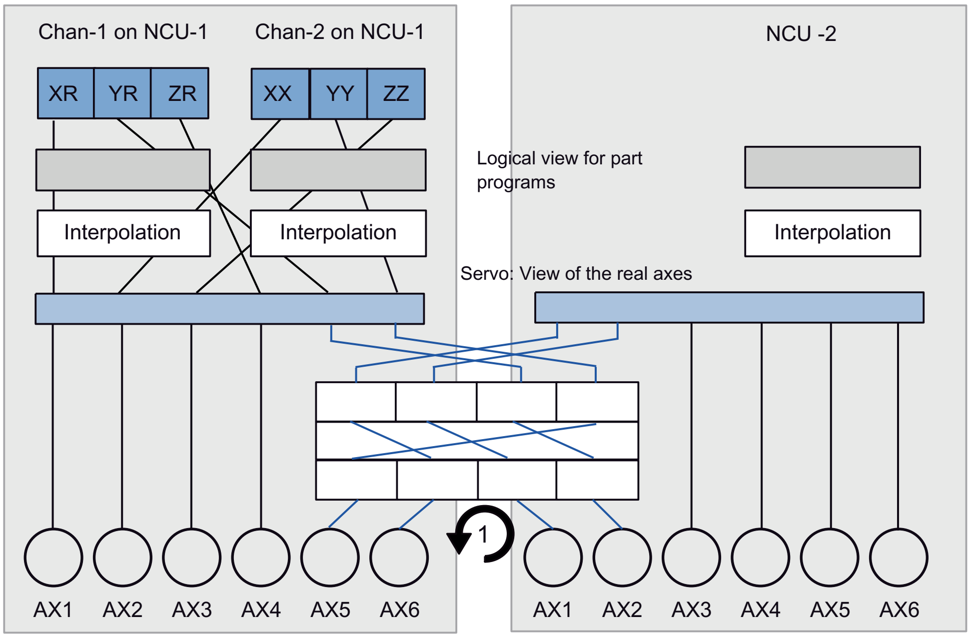


Figure 4-12 Configuration 3: NCU link with axis container in rotary state

Machine data of Configuration 2

```
; ########## NCU1 ##########
$MN_NCU_LINKNO = 1
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_LINK_NUM_OF_MODULES = 2
$MN_MM_SERVO_FIFO_SIZE = 3

$MN_ASSIGN_CHAN_TO_MODE_GROUP[1]=1

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "NC1_AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "NC1_AX3"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[3] = "NC1_AX4"
$MN_AXCONF_LOGIC_MACHAX_TAB[4] = "CT1_SL3"
$MN_AXCONF_LOGIC_MACHAX_TAB[5] = "CT1_SL4"

$MN_AXCT_AXCONF_ASSIGN_TAB1[0] = "NC1_AX5"
$MN_AXCT_AXCONF_ASSIGN_TAB1[1] = "NC1_AX6"
$MN_AXCT_AXCONF_ASSIGN_TAB1[2] = "NC2_AX1"
$MN_AXCT_AXCONF_ASSIGN_TAB1[3] = "NC2_AX2"

$SN_AXCT_SWWIDTH[0] = 1

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=5
$MC_AXCONF_MACHAX_USED[2]=4
$MC_AXCONF_MACHAX_USED[3]=0
$MC_AXCONF_MACHAX_USED[4]=0
$MC_AXCONF_MACHAX_USED[5]=0
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XR"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YR"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZR"
```

```
CHANDATA(2)
$MC_REFP_NC_START_LOCK=0
$MC_AXCONF_MACHAX_USED[0]=2
$MC_AXCONF_MACHAX_USED[1]=6
$MC_AXCONF_MACHAX_USED[2]=3
$MC_AXCONF_MACHAX_USED[3]=0
$MC_AXCONF_MACHAX_USED[4]=0
$MC_AXCONF_MACHAX_USED[5]=0
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YY"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZZ"
M30

; ########## NCU-2 ##########
$MN_NCU_LINKNO = 2
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_LINK_NUM_OF_MODULES = 2
$MN_MM_SERVO_FIFO_SIZE = 3

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "CT1_SL1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "CT1_SL2"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX3"
$MN_AXCONF_LOGIC_MACHAX_TAB[3] = "NC2_AX4"
$MN_AXCONF_LOGIC_MACHAX_TAB[4] = "NC2_AX5"
$MN_AXCONF_LOGIC_MACHAX_TAB[5] = "NC2_AX6"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=2
$MC_AXCONF_MACHAX_USED[2]=3
$MC_AXCONF_MACHAX_USED[3]=4
$MC_AXCONF_MACHAX_USED[4]=5
$MC_AXCONF_MACHAX_USED[5]=6
$MC_AXCONF_MACHAX_USED[6]=0
M30
```

4.4.6 Special features of interpolatory compensation

Measurement

The "Measurement" function supplies the compensated actual positions (ideal machine) required by the machine operator or programmer.

TEACH IN

The "TEACH IN" function also uses compensated position values to determine the actual positions to be stored.

Software limit switch

The ideal position values (i.e. the position actual values corrected by the MSEC and backlash compensation functions) are also monitored by the software limit switches.

Position display

The position actual-value display in the machine coordinate system shows the ideal (programmed) actual position value of the axis (ideal machine).

The position actual value determined by the measuring system plus the sum of MSEC and backlash compensation (= position actual value, measuring system 1/2) is displayed the "axis/spindle" service display (operating area "Diagnosis")

Compensation value display

The following compensation values are also output in the "Axis/spindle" service display (operating area "Diagnosis"):

- Absolute compensation value measuring system 1 or 2

  Displayed value corresponds to the total compensation value calculated from MSEC and backlash compensation associated with the actual position of the axis (measuring system 1 or 2).
- Compensation, sag + temperature

  Display value is the sum of the compensation values from sag compensation and temperature compensation for the actual position of the axis.

Reference point loss

If the reference point of the basic axis is lost (DB31, ... DBX60.4 or 60.5 = 0), then MSEC and sag compensation functions are deactivated in the axes involved. These are automatically reactivated when the reference point is reached.

Access protection

Currently there is no protection against access to the compensation tables.

Setting servo enables

As a result of the compensation relationship, a traversing movement by the base axis may also cause the compensation axis to move, making it necessary for controller enable signals to be set for these axes (PLC user program). Otherwise the compensation only has a limited effect.

Traversing signal output

The traversing signals in the compensation axis are output every time the compensation function is switched on/off and every time the number of active compensation tables changes.

Any change in the compensation value caused by the base axis motion does not result in output of traversing signals in the compensation axis.

4.5 Dynamic feedforward control (following error compensation)

4.5.1 General properties

Axial following error

The remaining system deviation of the position controller when traversing a machining axis is known as axial following error. Expressed in another way, the axial following error is the difference between the setpoint position and the actual position of the machine axis.

Effects

Particularly during acceleration in contour curvatures, e.g. circles and corners, this following error leads to undesirable, velocity-dependent contour violations.

Compensation

The axial following error can be reduced almost to zero with the help of the "dynamic feedforward control". The function is therefore also called "following error compensation".

Methods

There are two "dynamic feedforward control" methods:

- Speed feedforward control (velocity-dependent)
- Torque feedforward control (acceleration-dependent)

Activation

The feedforward control method is selected and activated using the machine data:

MD32620 $MA_FFW_MODE (feedforward control mode)

Value	Meaning
0	No feedforward control
1	Speed feedforward control with PT1 balancing
2	Torque feedforward control with PT1 balancing
3	Speed feedforward control with Tt balancing
4	Torque feedforward control with Tt balancing

Activation/deactivation in part program

The following axis-specific machine data can be used to define that the feedforward control for the respective axis/spindle can be activated and deactivated by the part program:

MD32630 $MA_FFW_ACTIVATION_MODE (activate feedforward control from program)

Value	Meaning
0	The feedforward control cannot be activated and deactivated from the part program. This means that the state specified using MD32620 $MA_FFW_MODE is always effective for the axis/spindle.
1	The feedforward control can be activated and deactivated from the part program. The operation becomes active immediately.
2	The feedforward control can be activated and deactivated from the part program. The operation only becomes active the next time that the axis comes to a standstill.

The feedforward control is activated/deactivated from the part program using the operations:

`FFWON`: Feedforward control ON

`FFWOF`: Feedforward control OFF

The default setting (i.e. M30 even after reset) is entered using the channel-specific machine data:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

`FFWON`/`FFWOF` is active for all axes/spindles in the axis mode, where:

MD32630 $MA_FFW_ACTIVATION_MODE = 1 (or 2)

and

MD32620 $MA_FFW_MODE = 1, 2, 3 or 4

The identical MD32630 setting should be used for axes that interpolate with each other.

The feedforward control should only be activated or deactivated while the axis/spindle is stationary in the axis mode, in order to prevent jerky motion. Hence the switchover is delayed automatically up to the next standstill through block search stop.

Note

A preprocessing stop has no effect for command or PLC axes traversing asynchronously to the part program processing. To ensure that `FFWON/FFWOF` only has an effect on the axis/spindle when it is next stationary in the axis mode, you must explicitly set MD32630 = 2 for each axis/spindle in the axis mode (see also "Forward feed control for command- and PLC axes (Page 284)").

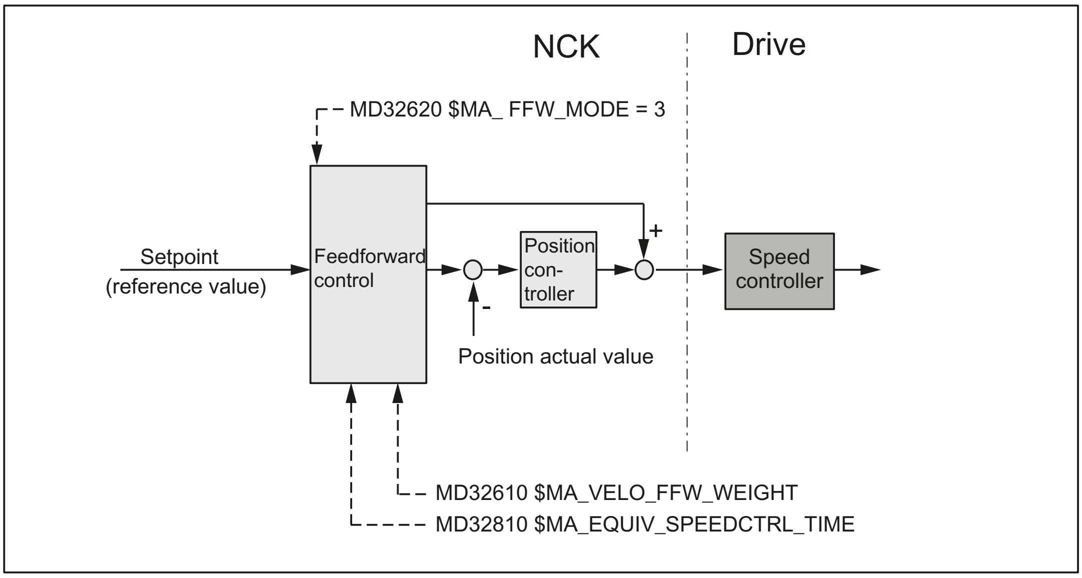
4.5.2 Speed feedforward control

Function

In the case of speed feedforward control, a velocity setpoint is also applied directly to the input of the speed controller. With this value the following error can be reduced to nearly zero (i.e. system deviation is 0) when the velocity is constant.

Commissioning

The following axis-specific parameters must be defined for the speed feedforward control during commissioning:



Equivalent time constant of the speed control loop (MD32810)

The equivalent time constant of the speed control loop must be determined accurately (e.g. graphically from a speed setpoint step response) and entered into the following machine data to correctly set the speed feedforward control:

MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

Feedforward control factor for speed feedforward control (MD32610)

The additional velocity setpoint can be weighted using a factor:

MD32610 $MA_VELO_FFW_WEIGHT

Range of values: 0 ... 1

"0" means: no feedforward control. As standard, the factor has a value of 1 (≙ 100%).

The factor should remain set at 100%, as this value is the optimum setting for an optimally set control loop for the axis/spindle as well as a precisely determined equivalent time constant of the speed control loop.

Fine adjustment

The speed feedforward control for the particular axis/spindle can be optimized by making slight changes (fine tuning) to the equivalent time constants of the speed control loop (MD32810).

To make this check, the axis/spindle should be traversed at a constant velocity and in the service display "Axis/spindle", the "System deviation" should be checked.

A small acceleration and a high feedrate should be chosen so that the values can be easily read on the service display. This produces very long acceleration phases from which it is easy to read off the control deviation.

Example:

MD32300 $MA_MAX_AX_ACCEL = 0.1	;	Maximum axis acceleration = 0.1 m/s²
MD32000 $MA_MAX_AX_VELO = 20000.0	;	Maximum axis velocity = 20000.0 mm/min

```
; Part program for setting the equivalent time constant
G1 F20000
FFWON
LOOP:
X1000
X0
GOTOB LOOP
M30
```

References

For detailed information about setting the equivalent time constants of the speed control loop (MD32810) refer to:

- Function Manual, Basic Functions; Velocities, Setpoint / Actual Value Systems, Closed-Loop Control (G2), Section: Optimization of the control

4.5.3 Torque feedforward control

Function

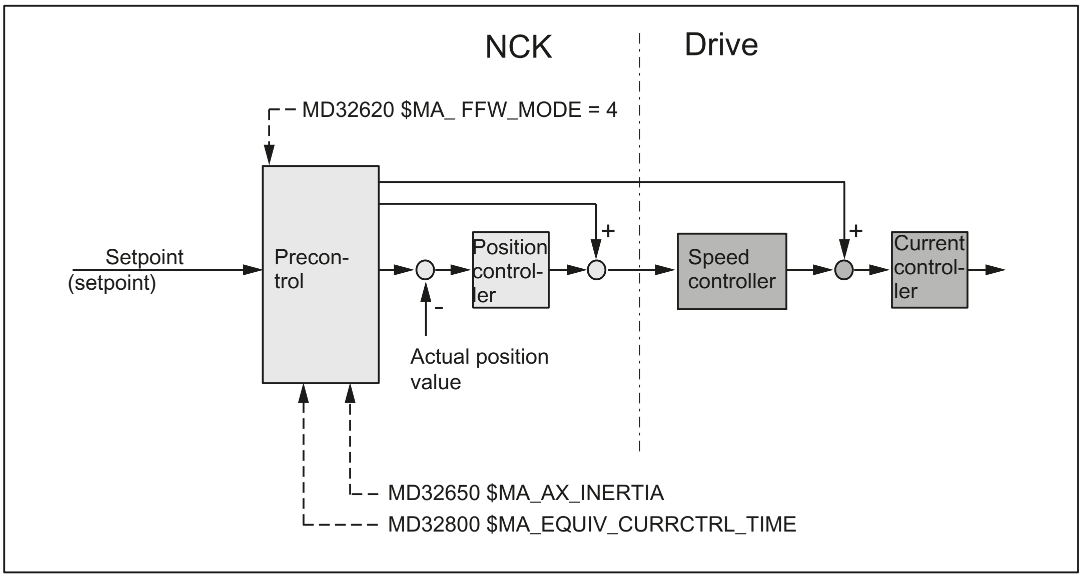
In the case of torque feedforward control, an additional current setpoint proportional to the torque is applied directly to the current controller input. This value is formed using the acceleration and moment of inertia.

Application

Torque feedforward control is required to achieve high contour accuracy where the demands on the dynamic response are very high. If set correctly, the following error can almost be completely compensated even during high acceleration.

Commissioning

The following axis-specific parameters must be defined during commissioning for torque feedforward control:



Equivalent time constant of the current control loop (MD32800)

The equivalent time constant of the current control loop must be determined accurately (e.g. graphically from the step response of the current control loop) and entered in the following machine data in order to correctly set the torque feedforward control:

MD32800 $MA_EQUIV_CURRCTRL_TIME (equivalent time constant current control loop for feedforward control)

Total moment of inertia of axis (MD32650)

The total moment of inertia (moment of inertia of drive plus load referred to the motor shaft) of the axis must be determined and entered in the following machine data:

MD32650 $MA_AX_INERTIA (inertia for torque feedforward control)

Fine adjustment

The torque feedforward control for the particular axis/spindle can be optimized by making slight changes (fine tuning) to the values in MD32800 and MD32650.

To make a check, the following error should be recorded via the trace functionality. In addition to traversing at a constant velocity, the following error should be monitored especially when the axis/spindle accelerates.

Note

As a result of the extremely fast sequences when accelerating, when commissioning the torque feedforward control, the service display cannot be used to check the fine adjustment.

References

For detailed information about setting the equivalent time constants of the current control loop (MD32810) refer to:

- Function Manual, Basic Functions; Velocities, Setpoint / Actual Value Systems, Closed-Loop Control (G2), Section: Optimization of the control

4.5.4 Dynamic response adaptation

Function

For axes that interpolate with one another, but with different axial control loop response times, dynamic response adaptation can be used to achieve identical time responses of all axes to ensure optimum contour accuracy without loss of control quality.

Commissioning

Time constant for dynamic response adaptation (MD32910)

The difference between the equivalent time constants of the "slowest" speed or current control loop and the particular axis should be entered as time constant for the dynamic response adaptation in the following machine data.

MD32910 $MA_DYN_MATCH_TIME (time constant of dynamic response adaptation)

Example:

Equivalent time constants of the speed control loop (MD32810) for active speed feedforward control of axes 1, 2 and 3:

- Axis 1: 2 ms
- Axis 2: 4 ms (dynamically the slowest axis)
- Axis 3: 1 ms

This means that the following values are obtained for the time constant of the dynamic response adaptation MD32910:

- Axis 1: 2 ms
- Axis 2: 0 ms
- Axis 3: 3 ms

Activation (MD32900)

The dynamic response adaptation is only active if the following machine data is set:

MD32900 $MA_DYN_MATCH_ENABLE= 1

Reference

Function Manual, Basic Functions; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2), Chapter: "Optimizing the closed-loop control"

4.5.5 Forward feed control for command- and PLC axes

Function

For command and PLC axes, it must be prevented that the feedforward control is activated/deactivated at higher velocities as follows:

MD32630 $MA_FFW_ACTIVATION_MODE = 2

With this setting, the `FFWON/FFWOF` operation only becomes active below the stationary velocity (MD36060 $MA_STANDSTILL_VELO_TOL) configured for this particular axis.

If the switchover instruction coincides with an axis motion, the required switchover is executed only in the next stoppage condition of the axis. This avoids the following error being suddenly established/reduced.

Note

A stoppage velocity set to a very high value can lead to the changeover of the feedforward control in the movement. Controls can be activated depending on the existing following error.

Commissioning

We recommend the following procedure when checking the feedforward control for command and PLC axes:

1. Check the stoppage velocity in MD36060.
2. Check the existing following error of the axis in stoppage condition.
3. Setting the changeover condition and activating it:

   MD32630 = 2
4. Traverse axis in the part program using the `POSA` operation.
5. Execute `FFWON` during the axis motion.
6. The K_v factor and following error displayed in the service display "Axis/spindle" must not jump.
7. A higher K_v factor and a lower following error are only obtained for traversing motion following standstill. However, the feedforward control is active only from the stoppage condition.

Essentially the same as when activating the feedforward control, for deactivation, the following applies:

1. Traverse axis in the part program using the `POSA` operation.
2. Execute `FFWOF` during the axis motion.
3. The K_v factor and following error displayed in the service display "Axis/spindle" must not jump.
4. A lower K_v factor and a higher following error are only obtained for traversing motion following standstill. However, the feedforward control is inactive only from the stoppage condition.

Example

In the following program example, axis A is traversed asynchronously to the path. An attempt is made to activate the feedforward control in the channel while traversing. Contrary to the geometry axes X, Y and Z, the feedforward control is not immediately effective for axis A. Here one waits for the stoppage after N60. Axis A then traverses with the feedforward control in N70.

Program code
N10 FFWOF
N20 POSA[A]=1000 FA[A]=10000
N30 G4 F1
N40 **FFWON**
N50 G0 X10 Y10 Z10
N60 WAITP(A)
N70 POSA[A]=1500 FA[A]=10000
N80 WAITP(A)
M30

4.5.6 Secondary conditions

Axes that are interpolating axes with one another

Also for axes that interpolate with one another, the feedforward control parameter should be optimally set **for each axis**, i.e. also several axes that are interpolating with one another can have different feedforward control parameters.

Check contour monitoring

As the two equivalent time constants:

- MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

and

- MD32800 $MA_EQUIV_CURRCTRL_TIME) (equivalent time constant current control loop for feedforward control)

also influence the contour monitoring, this should be subsequently checked.

Reference:
Function Manual, Basic Functions; Axis Monitoring, Protection Zones (A3)

Effect on servo gain factor

When the feedforward control is set correctly, the response to setpoint changes in the controlled system under speed feedforward control is as dynamic as that of the speed control loop or, under torque feedforward control, as that of the current control loop, i.e. the servo gain factor entered into MD32200 $MA_POS_CTRLGAIN hardly has any effect on the control behavior (e.g. corner errors, overshoots, circle/radius errors).

On the other hand, feedforward control does not affect the response to disturbances (synchronism). In this case, the servo gain factor entered in MD32200 is the active factor.

Service display "Servo gain factor"

When a feedforward control is active, the servo gain of the axis (corresponds to servo gain factor active as response to setpoint changes) shown in the service display "axis/spindle" is very high.

4.6 Friction compensation (quadrant error compensation)

4.6.1 General function description

In addition to the mass inertia and the machining forces, the frictional forces in the gearing and guideways of the machine influence the behavior of a machine axis. During the acceleration of an axis from standstill, especially the transition from static friction to the significantly smaller sliding friction has a negative affect with regard to the contour accuracy.

The sudden change in the friction force results in a briefly increased following error. With interpolating axes (path axes), this results in significant contour violations. For circles, the contour violations occur especially at the quadrant transitions due to the standstill of one of the involved axis at the direction reversal.

Therefore, an additional setpoint pulse is injected as a compensation value for this friction or quadrant error compensation when the axis accelerates from standstill, i.e. at the transition from static to sliding friction. In this way, contour violations can be almost completely avoided at the quadrant transitions of circular contours.

Acceleration-dependent friction compensation

In most cases, a compensation value independent of the axial acceleration with constant amplitude is sufficient for the quadrant error compensation. However, if the compensation value is dependent on the acceleration, an adaptation characteristic can be activated via the "friction compensation with adaptation" in order to model this behavior.

Circularity test

The easiest way to commission the friction compensation is with the circularity test integrated in the user interface. A circle is traversed and the circular contour generated on the machine based on the actual position values of the involved machine axes and the deviations to the programmed ideal circular contour, especially at the quadrant transitions, displayed in graphical form.

The circularity test can be found on the user interface under:

SINUMERIK Operate

"Operating area switchover" > "Commissioning" > "Optimization/test" > "Circularity test"

4.6.2 Supplementary conditions

Note

Switch off setpoint-related compensations

The following compensations affect the position setpoint and must be switched off before the measurement of the axes involved in the circularity test:

- Sag and angularity compensation (CEC):
  MD32710 $MA_CEC_ENABLE[<axis>] = 0
- Direction-dependent leadscrew error compensation:
  MD32710 $MA_CEC_ENABLE[<axis>] = 0
- Temperature compensation:
  MD32750 $MA_TEMP_COMP_TYPE[<axis>] = 0

4.6.3 Friction compensation with a constant compensation value

4.6.3.1 Function activation

Enable

The general enabling of the friction compensation is via:

MD32490 $MA_FRICT_COMP_MODE[<axis>] = 1

Activation

The activation of the friction compensation with constant compensation value is via:

- MD32500 FRICT_COMP_ENABLE[<axis>] = 1 (friction compensation ON)
- MD32510 $MA_FRICT_COMP_ADAPT_ENABLE[<axis>] = 0 (adaptation OFF)

Parameters

The following parameters are calculated for friction compensation with constant compensation value:

- MD32520 $MA_FRICT_COMP_CONST_MAX (maximum compensation value)
  For friction compensation with constant compensation value, the parameterized value is injected as compensation value.
- MD32540 $MA_FRICT_COMP_TIME (friction compensation time constant)
  The compensation value is injected via a DT1 filter. The compensation value decays according to the parameterized time constants.

4.6.3.2 commissioning

Circularity test

It is recommended that the circularity test be used for the commissioning of the friction compensation with constant injected value, as described above. The commissioning sequence is divided into the following steps:

1. Perform circularity test **without** friction compensation
2. Perform circularity test **with** friction compensation and initial parameter values
3. Perform circularity tests with friction compensation and modified parameter values
4. Complete circularity tests with friction compensation and optimized parameter values

Circularity test without friction compensation

A circularity test without friction compensation should be performed to determine the initial quality of the circular contour at the quadrant transitions. To do this, switch off the friction compensation temporarily:

MD32500 FRICT_COMP_ENABLE[<axis>] = 0

The following figure shows a typical example of quadrant transitions without friction compensation:

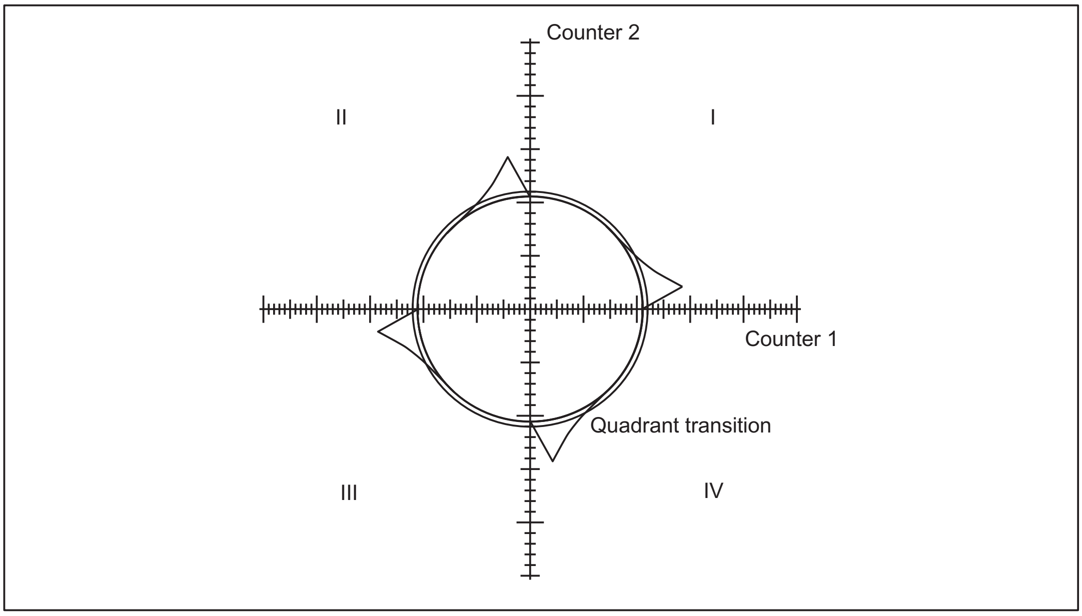


Figure 4-13 Quadrant transitions without friction compensation

Then switch on the friction compensation with constant compensation value:

MD32500 FRICT_COMP_ENABLE[<axis>] = 1

Circularity test with friction compensation and initial parameter values

It is recommended that a relatively small compensation value, as well as a time constant of just a few position control cycles, be set as initial parameter values, e.g.:

- MD32520 $MA_FRICT_COMP_CONST_MAX[<axis>] = 10 [mm/min]
- MD32540 $FRICT_COMP_TIME[<axis>] = 0.008 [ms]

The circularity test performed with these parameter values provides an initial assessment of the friction compensation.

Compensation value too small

Too small a compensation value (MD32520) in the circularity test is indicated by insufficient compensation of the contour deviations at the quadrant transitions.

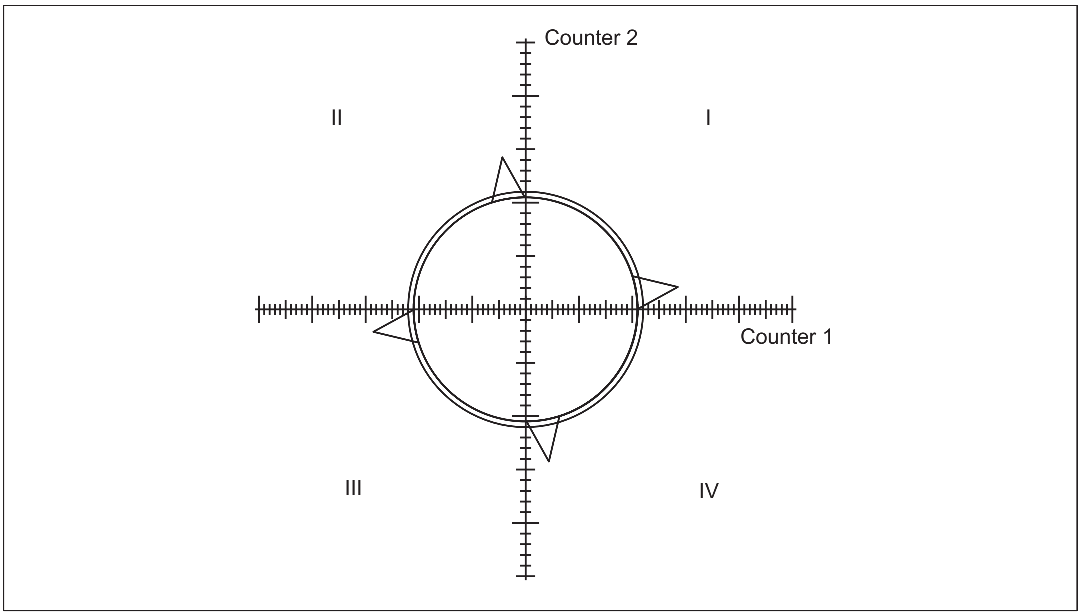


Figure 4-14 Compensation value set too small

Compensation value too large

Too large a compensation value (MD32520) in the circularity test is indicated by overcompensation of the contour deviations at the quadrant transitions.

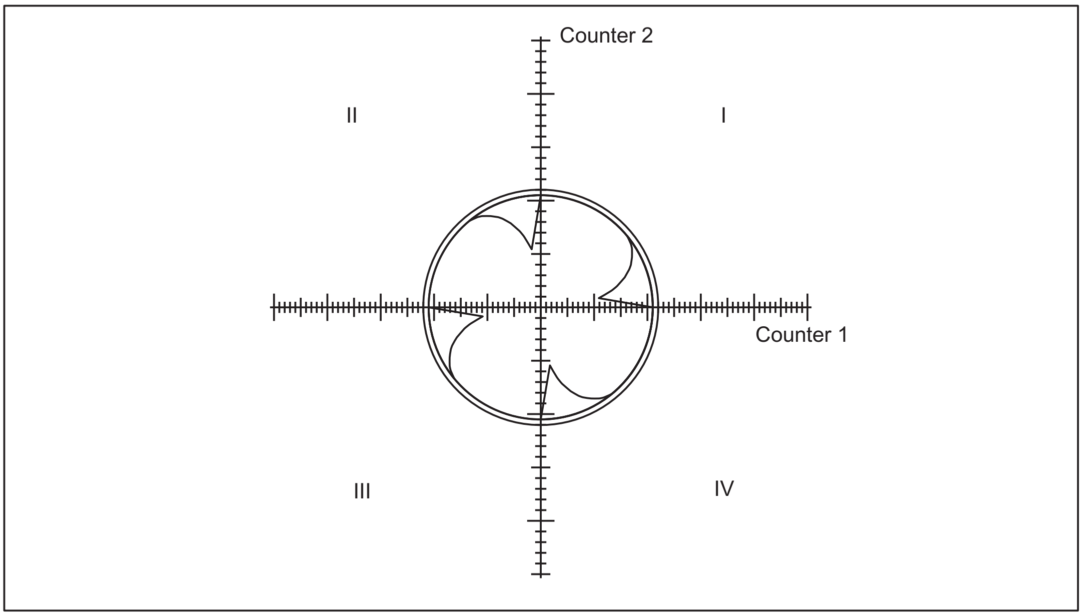


Figure 4-15 Compensation value set too large

Time constant too small

Too small a time constant (MD32540) in the circularity test is indicated by short-time compensation of the contour deviations at the quadrant transitions which immediately increase thereafter.

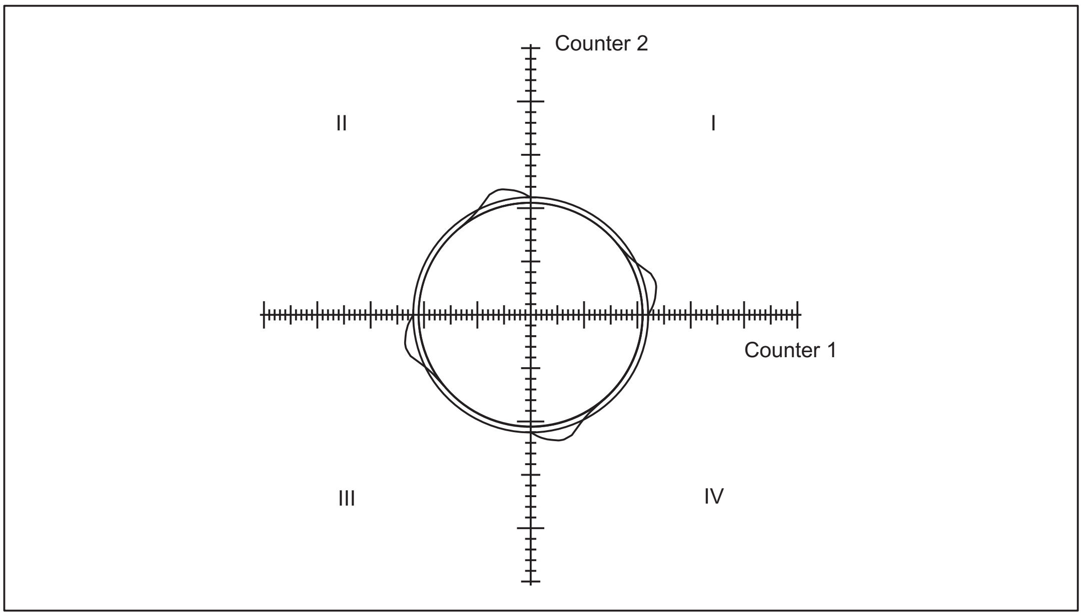


Figure 4-16 Compensation time constant set too small

Time constant too large

Too large a time constant (MD32540) in the circularity test compensates the contour deviations at the quadrant transitions. (Requirement: The optimum compensation value has already been determined.) However, with too large a time constant, the compensation value applies too long and results in an overcompensation at the next circular contour.

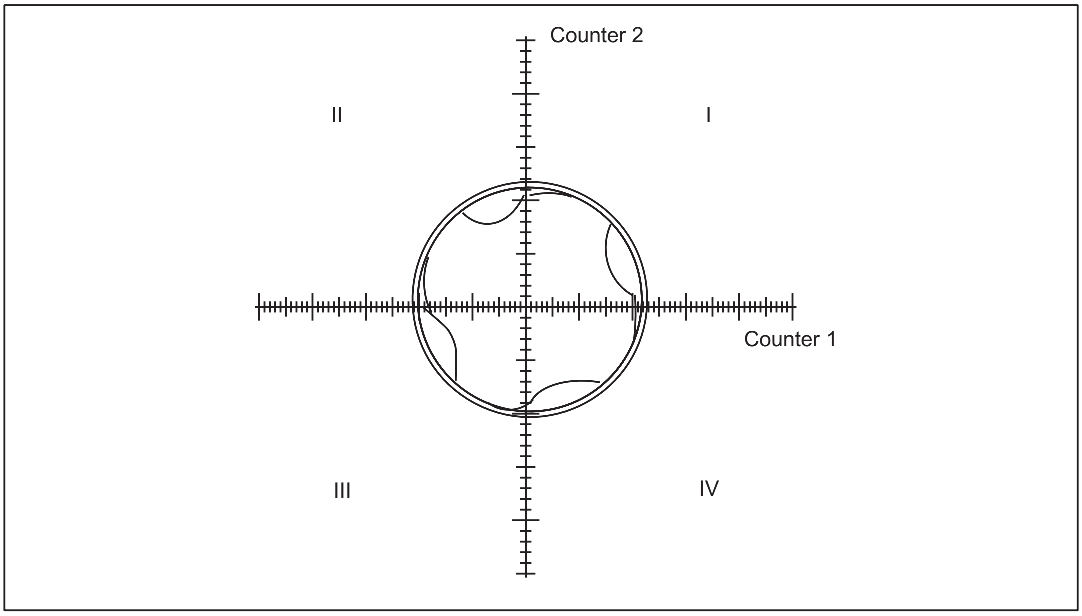


Figure 4-17 Time constant set too large

Optimization of the compensation parameters

To optimize the compensation parameters, the circularity test must be repeated several times and the values changed in small increments. It is recommended that the optimization be performed with different values for radius and path velocity that are typical for the machining operations performed on the machine.

Each effect of a parameter change should be checked with a subsequent circularity test and documented.

Mean value generation

If different parameter values result for different radii and path velocities, the best values should be determined via mean value generation.

Good friction compensation setting

With a good friction compensation setting, "no" contour violations can be detected at the quadrant transitions.

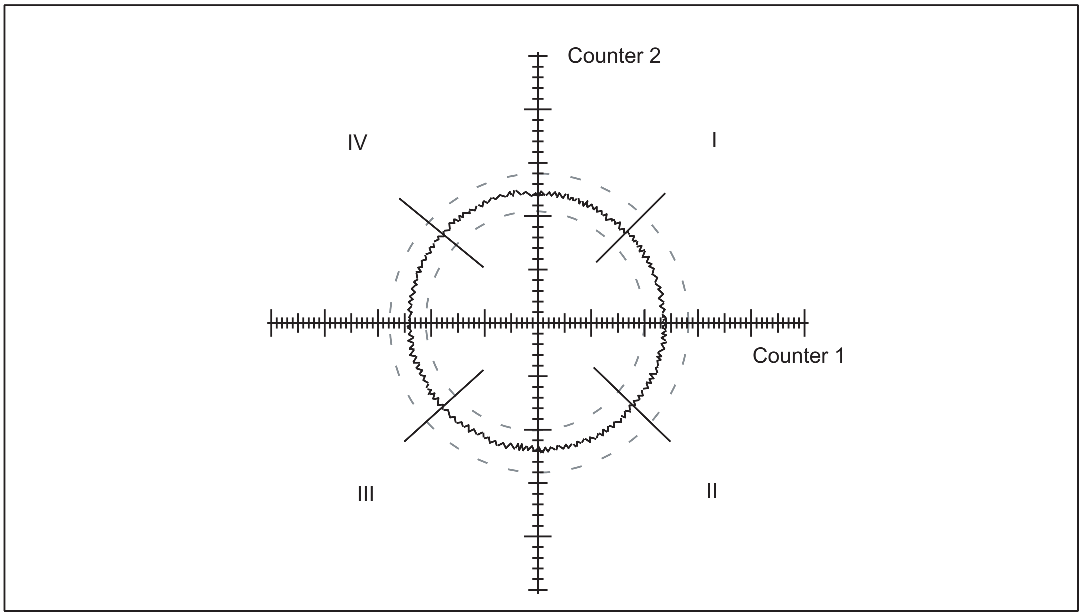


Figure 4-18 Good friction compensation setting

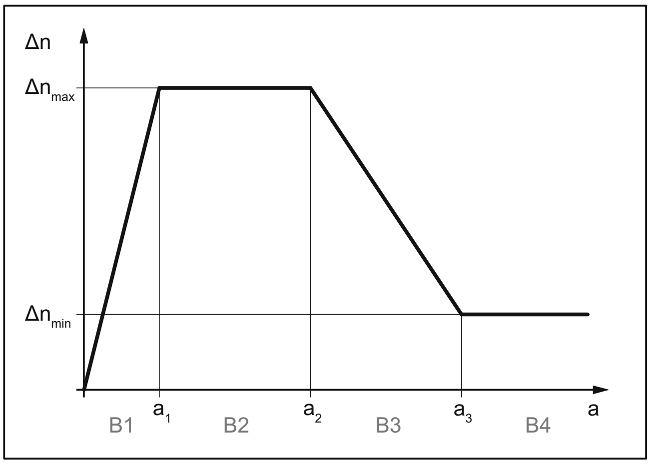
Acceleration-dependent compensation value

If the compensation value proves to be acceleration-dependent, the "friction compensation and adaptation" described below can be injected in a following step.

4.6.4 Friction compensation with acceleration-dependent compensation value

4.6.4.1 Description of functions

If the compensation value is highly dependent on the acceleration, normally a smaller compensation value must be injected for optimum compensation with larger accelerations than for smaller accelerations. This dependency can be modeled via the following adaptation characteristic.



Δn_{max}	Maximum compensation value
Δn_{min}	Minimum compensation value
a_1	Acceleration value 1
a_2	Acceleration value 2
a_3	Acceleration value 3
Bn	Acceleration range with n = 1, 2, ... 4
with:	Accelerations: $a_1 < a_2 < a_3$
	Compensation values: $\Delta n_{max} > \Delta n_{min}$, in special cases also $\Delta n_{max} < \Delta n_{min}$

The compensation value Δn is calculated according to the respective acceleration range B1 to B4 as follows:

Range	With acceleration a	⇒ Compensation value Δn
B1	$a < a_1$	$\Delta n = \Delta n_{max} * a / a_1$
B2	$a_1 \leq a \leq a_2$	$\Delta n = \Delta n_{max}$
B3	$a_2 < a < a_3$	$\Delta n = \Delta n_{max} + [(\Delta n_{min} - \Delta n_{max}) / (a_3 - a_2)] * (a - a_2)$
B4	$a \geq a_3$	$\Delta n = \Delta n_{min}$

4.6.4.2 Function activation

Enable

The general enabling of the friction compensation is via:

MD32490 $MA_FRICT_COMP_MODE[<axis>] = 1

Activation

The activation of the friction compensation with adaptation characteristic is performed via:

- MD32500 FRICT_COMP_ENABLE[<axis>] = 1 (friction compensation ON)
- MD32510 $MA_FRICT_COMP_ADAPT_ENABLE[<axis>] = 1 (adaptation characteristic OFF)

4.6.4.3 commissioning

To determine the characteristic parameters, the optimum compensation value Δn_{opt} must be determined at various operating points of the specified dynamic response range. See Section "commissioning (Page 289)". A sufficiently large number of measured values for large path velocities and small circle radii is particularly important.

For the evaluation of the determined value pairs, it is recommended that these are displayed graphically:
$\Delta n_{opt} = f(a)$, with Δn_{opt} = optimum compensation value and a = acceleration at the quadrant transitions.

The parameters of the adaptation characteristic determined from the measurement results must then be entered in the machine data.

Characteristic parameters

Acceleration values

The acceleration which arises at the quadrant transitions of the axis changing direction is calculated as follows:

$a = v^2 / r$, with v = path velocity and r = circle radius

Note

The path velocity and therefore the axial acceleration a can be varied simply via the feedrate override switch.

The acceleration values a_1, a_2 and a_3 determined as characteristic parameters must be entered in the following machine data. The following condition must be satisfied: $a_1 < a_2 < a_3$

- MD32550 $MA_FRICT_COMP_ACCEL1 (acceleration value 1)
- MD32560 $MA_FRICT_COMP_ACCEL2 (acceleration value 2)
- MD32570 $MA_FRICT_COMP_ACCEL3 (acceleration value 3)

Compensation values

The compensation values Δn_{min}, Δn_{max} determined as characteristic parameters must be entered in the following machine data:

- MD32520 $MA_FRICT_COMP_CONST_MAX (maximum compensation value)
- MD32530 $MA_FRICT_COMP_CONST_MIN (minimum compensation value)

Note

If satisfactory results cannot be obtained for very small path velocities, the computational resolution may have to be increased:

- MD10200 $MA_INT_INCR_ PER_MM (computational resolution for linear positions)
- MD10210 $MA_INT_INCR_PER_DEG (computational resolution for angular positions)

4.6.5 Compensation value for short traversing blocks

The compensation value determined for the quadrant error compensation can lead to overcompensation in short traversing blocks. The overcompensation can be avoided by reducing the compensation value specifically for traversing blocks that are traversed within one interpolation cycle. However, the size of the reduction is a value that must be determined empirically as it depends to a large extent for every axis on the particular situation at the machine. A percentage of the compensation value determined in the circularity test is set via the machine data:

MD32580 $MA_FRICT_COMP_INC_FACTOR (compensation value for short traversing blocks)

4.7 Measures for hanging (suspended axes)

4.7.1 Electronic counterweight

Axis without counterweight

For axes that have a weight load without counterweight, then after the brake is released, the hanging (suspended) axis drops and the following response is obtained:

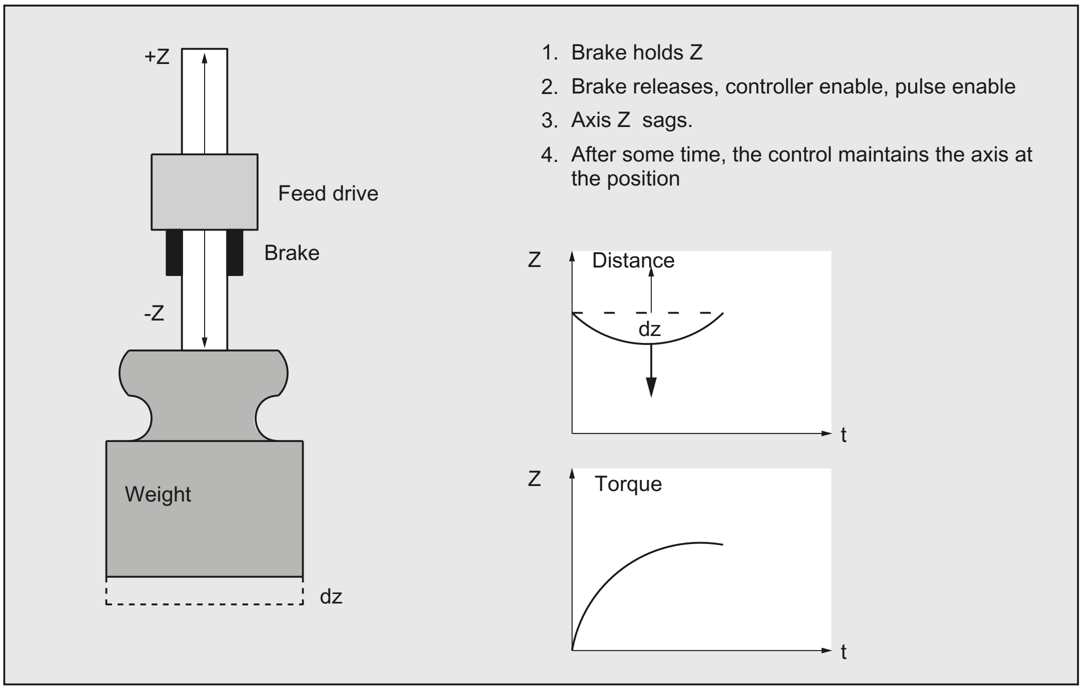


Figure 4-19 Drop of a hanging axis without counterweight

"Electronic counterweight" function

A hanging (suspended) axis can almost be completely prevented from dropping (sagging) using the "electronic counterweight" function.

The electronic counterweight prevents axes with a weight load from sagging when the closed-loop control is switched on. After releasing the brake, the constant counterweight torque maintains the position of the vertical axis.

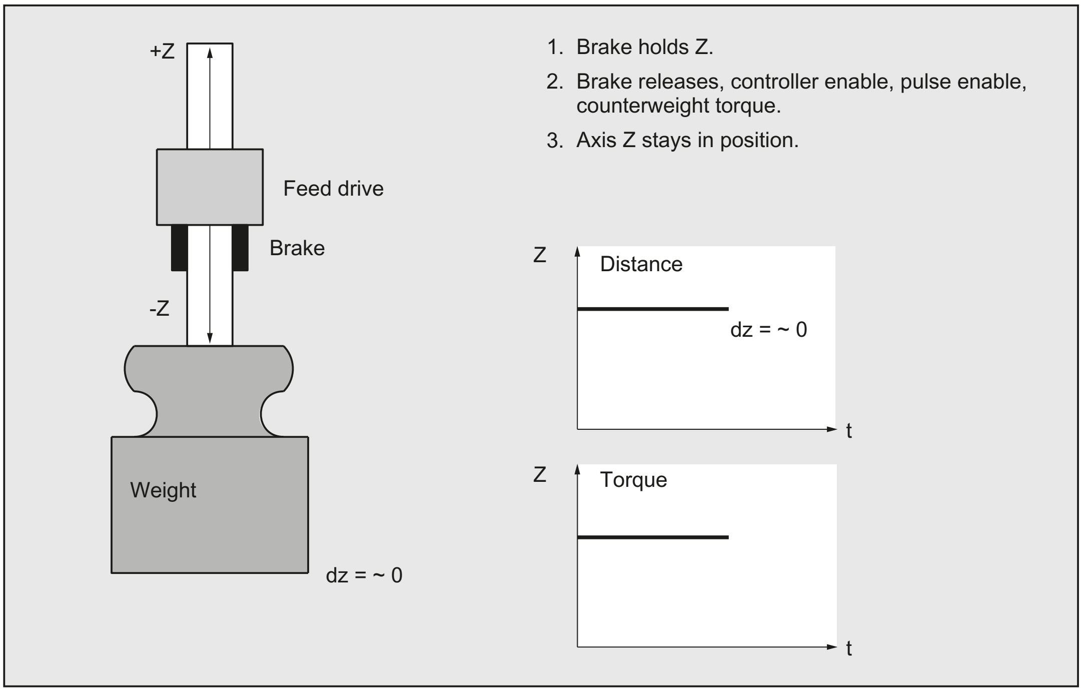


Figure 4-20 Lowering of a vertical axis with electronic weight compensation

Commissioning

Note

The "electronic counterweight" is commissioned through the drive!

Reference

For additional information, see the following:

SINAMICS S120 Function Manual Drive Functions

4.7.2 Reboot delay

Side-effect of a reboot from the user interface

The activation of the machine data, etc., from the user interface requires that the NCK is booted. This can cause hanging axes to drop somewhat. The "reboot delay" function can be used to avoid this.

Reboot delay

The reboot delay results in the NCK and PLC being shut down with a delay and communicates the pending shutdown in order to prevent hanging (suspended) axes from dropping.

Note

The reboot delay acts only for the controlled POWER ON from the user interface.

The reboot delay is not activated for a POWER FAIL (power failure) or a hardware reboot.

Reboot sequence

The command software can use the "_N_IBN_SS" PI service to initiate a reboot of the NCK and the PLC.

NCK immediately activates alarm 2900 in response to the PI service.

Mechanical axis brakes can be activated in the time that the NCK allows to expire from the PI service to the reboot (reboot delay time, refer to MD10088 $MN_REBOOT_DELAY_TIME).

Reactions to alarm 2900

- The following NC/PLC signals are **canceled**, i.e. set to zero:
  - DB11 DBX 6.3 (mode group ready) ; all mode groups
  - DB21, … DBX 36.5 (channel ready) ; all channels
  - DB31, … DBX 61.2 (axis ready) ; all axes
- The motor brakes along the current limit.

  For further details, see machine data:
  - MD36610 $MA_AX_EMERGENCY_STOP_TIME (braking ramp time when errors occur)
  - MD36620 $MA_SERVO_DISABLE_DELAY_TIME (switchoff delay controller release)

  Note

  After the shutdown delay, the NCK withdraws the controller enable (MD36620) of the position control.

- The following NC/PLC interface signals remain at 1:

  DB10 DBX108.7 (NC ready)

By using the machine data:
MD11410 $MN_SUPPRESS_ALARM_MASK (mask for suppressing special alarms) (BIT20) the alarm 2900 is suppressed, however, the NCK triggers the same reactions.

As alarm 2900 deactivates the axis position control, this alarm must be configured to initiate that the **mechanical brakes are closed by the PLC**. Rebooting the PLC forces the PLC outputs to change to defined zero. The brakes must be connected up in such a way that they **remain closed at zero**, i.e. a 1 signal on the PLC allows the brakes to open.

Note

In terms of its reactions, the alarm is the same as the Emergency Stop alarm (3000). For internal reasons, the reboot delay time of the NCK can be slightly increased.

Activation

The reboot delay can be activated as follows:

MD10088 $MN_REBOOT_DELAY_TIME (reboot delay) **> 0**

The value that has been entered supplies the reboot delay time in seconds.

Evaluation with a system variable

System variable $AN_REBOOT_DELAY_TIME can be read in a synchronized action. A value greater than zero indicates that the reboot request was initiated from the command software and how much time (in seconds) the NCK waits until reboot (POWER OFF and subsequent POWER ON). In a synchronized action, the user can identify the pending reboot and appropriately respond (e.g. with "Safe Standstill" for a Safety Integrated application). $AN_REBOOT_DELAY_TIME is 0.0 provided no reboot request from the command software is pending.

4.8 Data lists

4.8.1 Machine data

4.8.1.1 General machine data

Number	Identifier: $MN_	Description
10050	SYSCLOCK_CYCLE_TIME	Basic system clock cycle
10070	IPO_SYSCLOCK_TIME_RATIO	Factor for interpolator clock cycle
10082	CTRLOUT_LEAD_TIME	Shift of setpoint transfer time
10083	CTRLOUT_LEAD_TIME_MAX	Maximum permissible setting for shift of setpoint transfer time
10088	REBOOT_DELAY_TIME	Reboot delay
18342	MM_CEC_MAX_ POINTS[t]	Maximum number of interpolation points of sag compensation

4.8.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Reset G groups

4.8.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32450	BACKLASH	Backlash
32452	BACKLASH_FACTOR	Weighting factor for backlash
32456	BACKLASH_DYN	Compensation value for the dynamic backlash compensation
32457	BACKLASH_DYN_MAX_VELO	Limitation of the dynamic backlash compensation value change
32490	FRICT_COMP_MODE	Type of friction compensation
32500	FRICT_COMP_ENABLE	Friction compensation active
32510	FRICT_COMP_ADAPT_ENABLE	Friction compensation adaptation active
32520	FRICT_COMP_CONST_MAX	Maximum friction compensation value
32530	FRICT_COMP_CONST_MIN	Minimum friction compensation value
32540	FRICT_COMP_TIME	Friction compensation time constant
32550	FRICT_COMP_ACCEL1	Adaptation acceleration value 1

Number	Identifier: $MA_	Description
32560	FRICT_COMP_ACCEL2	Adaptation acceleration value 2
32570	FRICT_COMP_ACCEL3	Adaptation acceleration value 3
32580	FRICT_COMP_INC_FACTOR	Weighting factor for friction compensation value for short traversing motion
32610	VELO_FFW_WEIGHT	Feedforward control factor for velocity/speed feedforward control
32620	FFW_MODE	Feedforward control mode
32630	FFW_ACTIVATION_MODE	Activate feedforward control from program
32650	AX_INERTIA	Inertia for torque feedforward control
32700	ENC_COMP_ENABLE	Interpolatory compensation
32710	CEC_ENABLE	Enabling of sag compensation
32711	CEC_SCALING_SYSTEM_METRIC	System of units for sag compensation
32720	CEC_MAX_SUM	Maximum compensation value for sag compensation
32730	CEC_MAX_VELO	Change of velocity during sag compensation
32750	TEMP_COMP_TYPE	Temperature compensation type
32760	COMP_ADD_VELO_FACTOR	Velocity increase as a result of compensation
32711	CEC_SCALING_SYSTEM_METRIC	System of units for sag compensation
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
32910	DYN_MATCH_TIME	Time constant for dynamic response adaptation
36500	ENC_CHANGE_TOL	Maximum tolerance for position actual value switchover
38000	MM_ENC_COMP_MAX_POINTS	Number of interpolation points with interpolatory compensation

4.8.2 Setting data

4.8.2.1 General setting data

Number	Identifier: $SN_	Description
41300	CEC_TABLE_ENABLE[t]	Enable evaluation of beam sag compensation table
41310	CEC_TABLE_WEIGHT[t]	Weighting factor for beam sag compensation table

4.8.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43900	TEMP_COMP_ABS_VALUE	Position-independent temperature compensation value
43910	TEMP_COMP_SLOPE	Gradient for position-dependent temperature compensation
43920	TEMP_COMP_REF_POSITION	Reference position for position-dependent temperature compensation

4.8.3 Signals

4.8.3.1 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
NC Ready	DB10.DBX108.7	DB2700.DBX2.7

4.8.3.2 Signals from mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode group ready	DB11.DBX6.3	DB3100.DBX0.3

4.8.3.3 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Channel ready	DB21, ...DBX36.5	DB3300.DBX4.5

4.8.3.4 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate dynamic backlash compensation	DB31, ...DBX25.0	DB380x.DBX5001.0

4.8.3.5 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized 1	DB31, ...DBX60.4	DB390x.DBX0.4
Referenced/synchronized 2	DB31, ...DBX60.5	DB390x.DBX0.5
Axis ready	DB31, ...DBX61.2	DB390x.DBX1.2
Dynamic backlash compensation active	DB31, ...DBX102.0	DB390x.DBX5006.0

K5: Mode groups, channels, axis interchange 5

5.1 Brief description

Mode group

A mode group is a collection of machine axes, spindles and channels which are programmed to form a unit. In principle, a single mode group equates to an independent NC control (with several channels). A mode group is made up of those channels that always have to operate simultaneously in the same mode.

Note

There is one mode group as standard.

Reference:
Function Manual Basic Functions; Mode Group, Channel, Program Operation (K1)

Note

Only 1 mode group is available for SINUMERIK 828D.

Channels

Every channel has its own program decoding, block preparation and interpolation functions. A part program can be processed independently within a channel.

Note

There is one channel available as standard.

Reference:
Function Manual Basic Functions; Mode Group, Channel, Program Operation (K1)

The processes in several channels of a mode group can be synchronized in the parts programs.

Note

Only 1 channel is available for SINUMERIK 828D.

Axis/spindle interchange

After control system power ON, an axis/spindle is assigned to a specific channel and can only be utilized in the channel to which it is assigned.

With the function "Axis/spindle interchange" it is possible to enable an axis/spindle and to allocate it to another channel, that means to replace the axis/spindle.

Axis/spindle interchange can be activated via the parts program, via the PLC program and from motion-synchronous actions.

Axis/spindle interchange is also possible via:

- Programming in parts program GET/GETD.
- Automatically through programming of axis name.
- Without preprocessing stop and existing synchronization between preprocessing and main run.
- Through PLC via the VDI interface to the NCK.

Axis replacement extensions

- Set axis replacement behavior variable.
- Axis replacement with an axis container rotation with implicit GET/GETD.
- Axis replacement without pre-processing stop of axes not involved in the contour.
- Geometry axis with rotated frame (ROT) and axis replacement in JOG operational mode.
- Axis replacement via synchronized actions GET(axis), AXTOCHAN.

Note

For SINUMERIK 828D, an axis/spindle interchange is not possible between channels.

5.2 Mode groups - only 840D sl

Mode groups

A mode group combines NC channels with axes and spindles to form a machining unit.

A mode group contains the channels that are required to run simultaneously in the same mode from the point of view of the machining sequence.

Any axis can be programmed in any channel of a certain mode group. A mode group therefore corresponds to an independent, multiple-channel NC.

example

On large machine tools (machining centers), it may be necessary for a part program to be processed on one part of the machine while new workpieces to be machined need to be clamped and set up on another part. Such tasks usually require two independent NC controls.

With the mode group function, both tasks can be implemented on one NC control with two mode groups because a different mode can be set for each mode group (AUTOMATIC mode for the program processing, JOG for setting up a workpiece).

Mode group assignment

The configuration of a mode group defines the channels, geometry axes, machine axes and spindles which it is to contain.

A mode group consists of one or several channels which must not be assigned to any other mode group. Machine axes, geometry axes and special axes themselves are assigned to these channels. A machine axis can only be assigned to the channels of one mode group and can only traverse in this mode group.

A mode group is configured with the following data:

- Channel-specific machine data:

  MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP (channel valid in mode group)
- Configuration data of the channels

Note

For more information about the first mode group, please refer to:

References:

Function Manual, Basic Functions; Mode Group, Channel, Program Operation Mode (K1),

---

5.3 Channels - only 840D sl

Note

The terms Channel, Channel Configuration, Channel States, Effects of Commands/Signals, etc. is described for the first channel in:
Reference:
Function Manual Basic Functions; Mode Group, Channel, Program Operation (K1)

For all other channels, this information applies, too.

5.3.1 Channel synchronization (program coordination)

Function

For example, for double-slide machining or real-time actions, the possibility for the synchronization of the machining between channels must be present. The channels affected shall perform certain processing procedures time-matched.

To allow this machining, the relevant channels must be joined to form a synchronization group (mode group).

The channel synchronization is made only with the NC language.

Preconditions

The relevant channels must belong to the **same mode group**.

Programming

There are special statements (commands) for the channel synchronization. In each case, they are listed in one block.

Table 5- 1 Program coordination statements

Statement	Meaning	
INIT (<channel-no.>, <path specification>, <acknowledgement mode>)	Selection of a program for processing in a certain channel:	
	<channel no.>:	Number of channel
	<path specification>:	An absolute or relative path to the NC program
	<acknowledgement mode>:	Acknowledgement mode: **N** (without) or **S** (synchronous)
CLEAR (<program name>)	Deletion of a program by indicating the program name.	
START(<channel no.>, <channel no.>, ...)	Starting the selected programs in other channels.	
	<channel no.>, ...:	Enumeration of the channel numbers
WAITM (<marker no.>, <channel no.>, <channel no.>, ...)	Unconditional wait: When a WAITM() call is reached, the axes of the current channel are decelerated and a wait made in the other channels to be synchronized until the marker number specified in the call is reached. The group is synchronized when the other channels are also decelerated as they reach their WAITM() command. The synchronized channels then continue operation.	
	<marker no.>:	The tag number must be the same in all channels.
	<channel no.>, ...:	Enumeration of the channel numbers (the own channel does not need to be specified).

Statement	Meaning
WAITE (<channel no.>, <channel no.>, ...)	Waits for the end of program of the specified channels (current channel not specified).
WAITMC (<marker no.>, <channel no.>, <channel no.>, ...)	Conditional wait in path controlled operation for the specified wait marker from the specified channels. The current channel can be specified, but this is optional. When processing continues after the wait marks from the other channels in the group have arrived, the wait marks of these channels are deleted.
SETM (<marker no.>, (<marker no.>, ...)	Set the wait markers for conditional wait with WAITMC() for the channel specified in the SETM(). The channel thus declares its wait characteristics for the partner channels as fulfilled. The command can be activated in synchronized actions. Up to 10 marks (0-9) can be set using one command.
CLEARM (<marker no.>, (<marker no.>, ...)	Cleat the wait markers for conditional wait with WAITMC() for the channel specified in the CLEARM(). The channel thus declares to its partner channels that its wait characteristic is fulfilled. The command can be activated in synchronized actions. Up to 10 marks (0 - 9) can be deleted using one command.

Note

For further information about WAITMC and SETM, see Section "Channel synchronization: Conditional wait in path controlled operation (Page 311)".

Note

A maximum of 100 markers (marker 0 ... 99) are available in a multi-channel system.

A single-channel system only has marker 0.

Example: Unconditional wait with WAITM

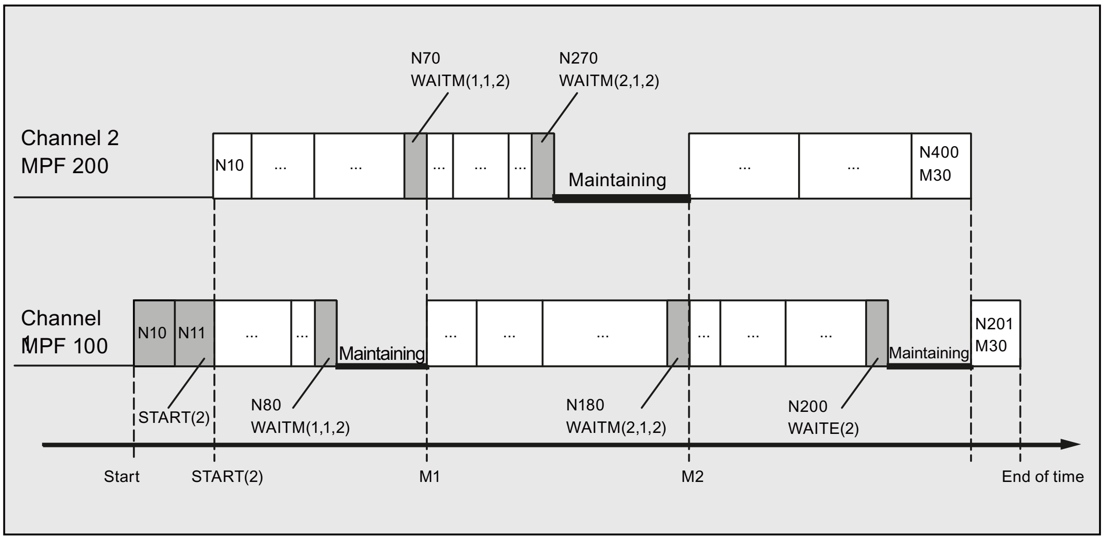
Channel 1: Program /_N_MPF_DIR/_N_MPF100_MPF is selected.

Program code	Comment
N10 INIT(2,"MPF200","N")	
N11 START(2)	
...	; Machining in channel 1
N80 WAITM(1,1,2)	; Wait until wait marker 1 is reached in channels 1 and 2.
...	; Additional machining in channel 1.
N180 WAITM(2,1,2)	; Wait until wait marker 2 is reached in channels 1 and 2.

Program code	Comment
...	; Additional machining in channel 1.
N200 WAITE(2)	; Wait for the end of program of channel 2
N201 M30	; End of program of channel 1, total end.

Channel 2: The INIT command (see N10 in _N_MPF100_MPF) selects the _N_MPF200_MPF program for execution in channel 2.

Program code	Comment
;$PATH=/_N_MPF_DIR	
...	; Machining in channel 2
N70 WAITM(1,1,2)	; Wait until wait marker 1 is reached in channels 1 and 2.
...	; Additional machining in channel 2.
N270 WAITM(2,1,2)	; Wait until wait marker 2 is reached in channels 1 and 2.
...	; Additional machining in channel 2.
N400 M30	; End of program in channel 2.



5.3.2 Channel synchronization: Conditional wait in path controlled operation

Function

For the conditional wait with WAITMC, deceleration and waiting is made only when not all channels to be coordinated have set their marker numbers for a synchronization.

The instants in time for generating wait marks and the conditional wait calls are decoupled.

Note

Markers can also be set for notification between channels when deceleration and waiting is not planned (no WAITMC command). In this case, the markers of the channel receive their values via Reset and NC Start.

Requirements

To use the conditional wait with WAITMC with reduced wait times:

- G64 path controlled operation must be set.
- The "LookAhead" function must be active.

Note

If exact stop (G60, G09) is selected, waiting with WAITMC() corresponds to waiting with WAITM().

Braking behavior

Starting with the motion block before the WAITMC() call, the wait markers of the other channels to be synchronized are checked. If these are already available, machining continues without braking (no wait):

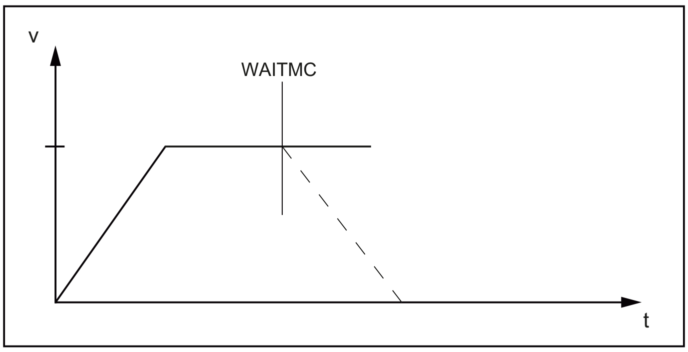


Figure 5-1 Change of the path velocity for the conditional waiting with WAITC: Wait markers for all channels already available

If the wait marker for a channel to be synchronized is missing, braking will be started. During braking, a check is made in each interpolation cycle whether the still missing wait markers for the channels to be synchronized have arrived in the meantime. In this case, a further acceleration to the path velocity is made and machining continues:

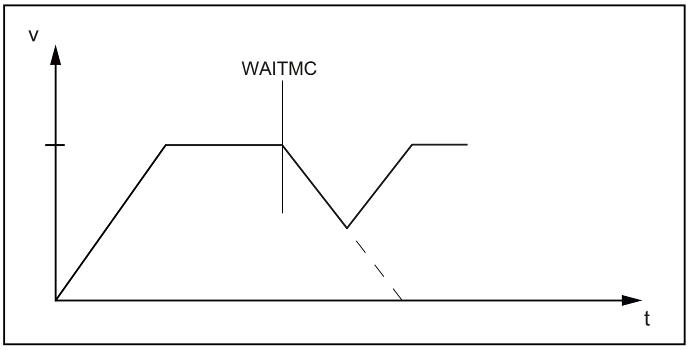


Figure 5-2 Change of the path velocity for the conditional waiting with WAITC: The last wait marker arrives during the braking

If the path velocity has been braked to zero before the expected markers of the channels to be synchronized have arrived, the machining stops until the missing markers have arrived. When the last expected marker arrives, acceleration resumes from standstill to the path velocity:

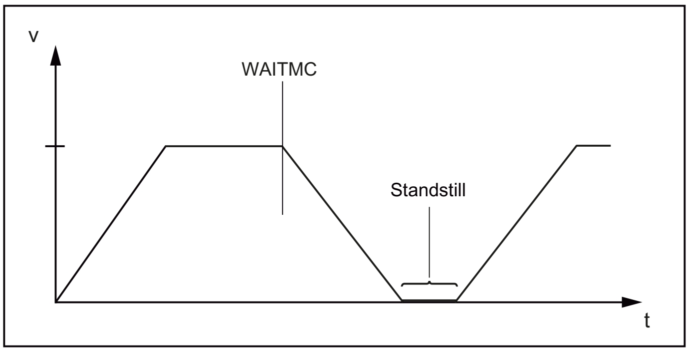


Figure 5-3 Change of the path velocity for the conditional waiting with WAITC: The last wait marker arrives after the braking

Block change in the braking ramp

If the IPOBRKA (block change possible in the braking ramp) motion criterion is active, the arrival of the wait marker causes an instantaneous switch to the next block and the axes started.

If the marker has not yet been reached or some other motion criterion prevents the block change, braking continues.

Example: Conditional wait in path controlled operation

The example is schematic and shows only those commands that are relevant to the synchronization process.

Channel 1:

Program code	Comment
%100	
N10 INIT(2, "_N_200_MPF","n")	; Select partner program channel 2.
N11 INIT(3,"_N_300_MPF","n")	; Select partner program channel 3.
N15 START(2,3)	; Start programs in channel 2, 3.
...	; Machining in channel 1.
N20 WAITMC(7,2,3)	; Conditional wait for marker 7 from channels 2 and 3.
...	; Further machining in channel 1.
N40 WAITMC(8,2)	; Conditional wait for marker 8 from channel 2.
...	; Further machining in channel 1.
N70 M30	; End channel 1.

Channel 2:

Program code	Comment
%200	
N200	; Machining in channel 2.
N210 SETM(7)	; Channel 2 sets wait marker 7.
...	; Further machining in channel 2.
N250 SETM(8)	; Channel 2 sets wait marker 8.
N260 M30	; End channel 2.

Channel 3:

Program code	Comment
%300	
N300	; Machining in channel 3.
...	
N350 WHEN <condition> DO SETM(7)	; Set wait marker in a synchronous action.
...	; Further machining in channel 3.
N360 M30	; End channel 3.

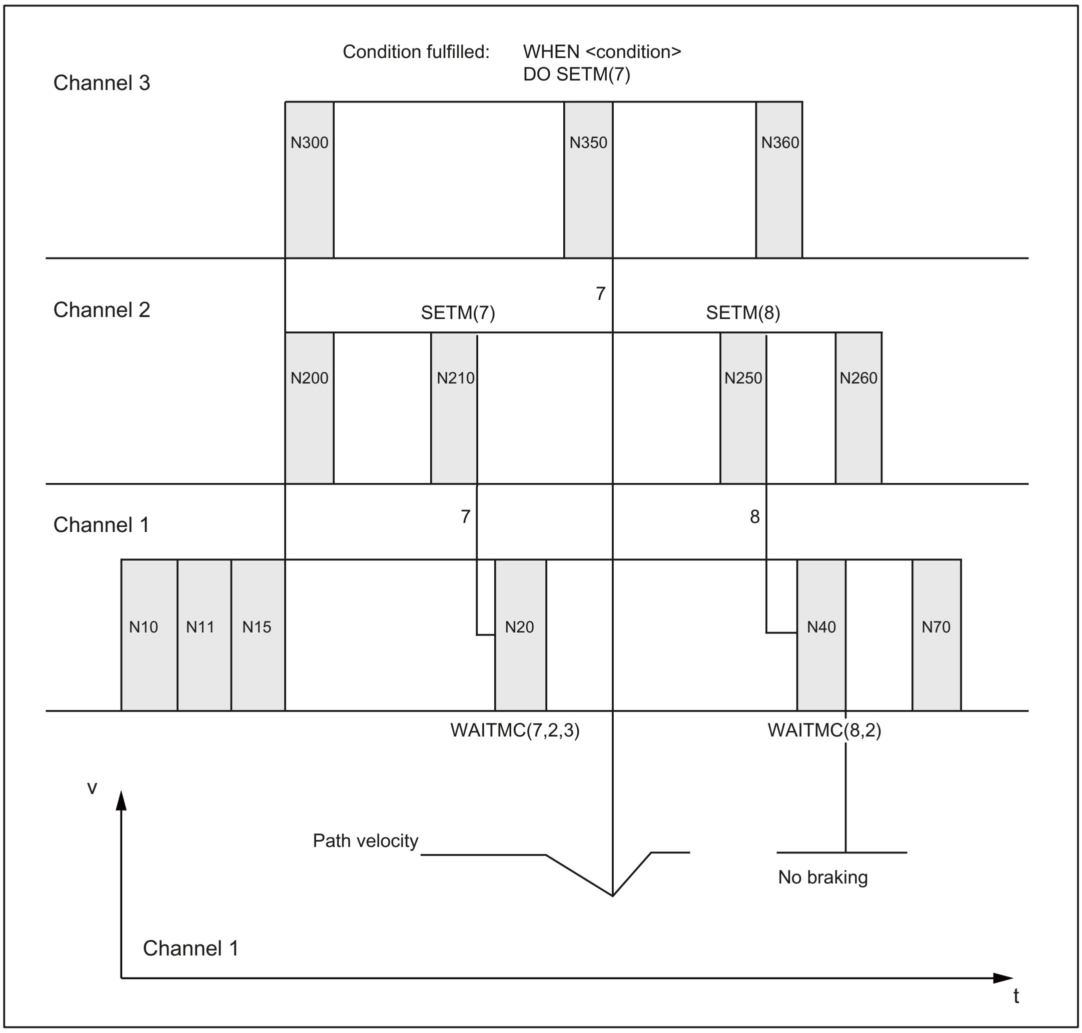


Figure 5-4 Conditional wait in path controlled operation with three involved channels (schematic)

Example: WAITMC and read-in disable

M555 is output in channel 3 while the axis is traversing and generates a read-in disabled (RID). Because the WAITMC is added to block N312, the wait marker is set and channel 2 continues to travel. The read-in disable causes the program processing in channel 3 to stop.

Note

For active G64, a WAITMC(...) block **does not create its own block** but is added to the previous block. A drop in velocity must be prevented when continuous-path mode is active. A WAITMC is therefore fulfilled if the preceding block is halted, e.g. by a read-in disable.

Channel 2:

Program code	Comment
N112 G18 G64 X200 Z200 F567	; Machining in channel 2.
N120 WAITMC(1,2,3)	; Conditional wait for marker 1 from channels 2 and 3.
...	; Further machining in channel 2 because the WAITMC is added to block N312.
...	; Further machining in channel 2.
N170 M30	; End channel 2.

Channel 3:

	Comment
	; During travel, M555 read-in disable.
N300	; Machining in channel 3.
N312 G18 G64 D1 X180 Z300 M555	
N320 WAITMC(1,2,3)	; Wait because of ELSP.

5.3.3 Running-in channel-by-channel

Function

For multi-channel systems, using the function "channel-by-channel running-in", the part program of a selected channel can be actually tested at the machine. The other channels are then in the "program test" state. This means that when starting all channels, only the axes of the selected channel are actually traversed.

In addition, for individual axes/spindles, whose channel is in the "program test" state, users have the option of suppressing the "program test" state and actually operating these axes/spindles together with those of the selected channel.

Advantage

For multi-channel systems, generating part programs places high demands on the ability of the programming engineer to think in abstract terms. By using the function "channel-by-channel running-in", testing these part programs can be more selectively arranged with less associated risk.

Application

The function "channel-by-channel running-in" is used for:

- Multi-channel systems
- Machines with POSA or command axis motion

Sequence

Multi-channel systems are either started at the same time or staggered in time, channel for channel. As an alternative, the PLC can start a channel and its part program initializes and starts the channels. The function "channel-by-channel running-in" supports both of these versions. As a consequence, a channel group is obtained, which should operate together. Normally, this group comprises all of the channels of the NCK being used.

Generally, a channel moves a tool adapter in the machining space and therefore the tool. Several channels each move a tool in the same machining space and therefore require (mandatory) synchronization between the channels in order to avoid collisions and to organize how they operate with one another. The following synchronizations are conceivable:

- Channel coordination via the part program commands `WAITM`, `WAITMC`, `WAITE`, `START`.
- Channel synchronization via the PLC.

  Example:

  In channel 1, M107 is kept with the read-in disable until channel 2 has reached M207 and vice versa.
- Axis interchange, i.e. a channel waits until the other channel relinquishes the axis.
- Freely programmed synchronization using global variables in the part program.
- Cross-channel couplings
- Axis container rotation
- Testing the program including the parallel synchronized actions in the main run and synchronization of the synchronized actions with the channel.

Under these general conditions, it is almost impossible to just start one channel - it would remain stationary at the first synchronization location.

Using the function "channel-by-channel running-in" all channels of the group are to be started, and only a few channels, generally just one channel, actually physically moves its axes. The other channels are then in the "program test" state.

This is the reason why users must define the channels in which they do not want any motion. This is made from the user interface in the "Program controls" menu. When selected, the following channel-specific signal is set in the HMI/PLC interface:

DB21, ... DBX25.7 (program test selected)

The activation is then made using the channel-specific NC/PLC interface signal:

DB21, ... DBX1.7 (activate program test)

The following interface signal is set in the PLC as checkback signal:

DB21, ... DBX33.7 (program test active)

Further, for a successful operation, it may be necessary that several axes/spindles - especially spindles - are actually physically operated, although their channel is in the "program test" state. The following NC/PLC interface signal is used for this purpose:

DB31, ... DBX14.0 (suppress program test)

Example:

A system comprises a main spindle and counterspindle. Two slides can operate on both the main spindle and counterspindle. Each slide is controlled from a separate channel. The main spindle is in channel 1, the counterspindle in channel 2. Channel 1 is tested and channel 2 is disabled using the channel-specific NC/PLC interface signal DB21, ... DBX1.7 (activate program test). The two workpiece spindles - main spindle and counterspindle - play somewhat of a "special role". A workpiece can be machined, without having to absolutely traverse the workpiece spindle in real terms in the channel. This is the reason why it is necessary that both workpiece spindles or both workpiece spindle aggregates actually move in real terms (where relevant, including axes at the workpiece).

Note

The "program test" state can only be activated/deactivated in the stopped channel state. However, the axis-specific NC/PLC interface signal "suppress program test" can always be activated.

Diagnostics

The "program test" state can be interrogated using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variables:

$AC_ISTEST	"Program test" state for the channel Supplies TRUE (1), if the "program test" state for the channel is active.
$AA_ISTEST[<n>]	"Program test" state for the axis <n> Supplies TRUE (1), if the "program test" state for axis <n> is active.

- Without preprocessing stop in the part program via the system variable:

$P_ISTEST	Supplies TRUE (1), if the "program test" state for the channel is active.

Example:

The channel runs under "program test" and axis "C" was withdrawn using "suppress program test". A query using system variables then supplies the following result:

$AC_ISTEST = TRUE

$P_ISTEST = TRUE

$AA_ISTEST[C] = FALSE

Boundary conditions

Axis replacement

The function "axis interchange" allows that an axis/spindle is known in several channels and can be programmed from this alternating (see Section "Axis/spindle replacement (Page 320)").

In conjunction with the functions "program test" and "channel-by-channel running-in", the following must be observed for an axis interchange:

- If only one of the channels is in the "program test" state, then the interchanged axis is taken from this channel and is inserted in a channel that is not in the "program test" state. For an interchanged axis with active axis disable, for a change via the channels with/without channel state "program test", then the state in the axis itself does not change (see example 3).
- For a program test, for end of part program/reset, for all axes/spindles that do not interpolate, resynchronization is made at the actual servo position. As a consequence, for an axis interchange that is first made after the end of the program, as the axis may only exit the channel at the end of the program, the simulated position reached is not transferred to the accepting channel.

Note

The programs should also include a WAIT tag at the end in order that they are simultaneously exited.

Examples

Example 1: Channel 2 is to be tested in a 3-channel system.

The following program test sequences are possible:

a) Program test without SERUPRO

1. The user decides which axes/spindles should actually be physically traversed. "Suppress program test" is set for these axes.
2. The "program test" state is selected for channels 1 and 3.
3. Channels 1, 2 and 3 are started via the PLC.
4. "Program test" can be selected again after the end of the program.
5. If the actual setting of "suppress program test" is also practical for other situations (channel 1 or channel 3 are to be tested), then this signal can remain set. This is certainly practical in many cases.

b) Program test with SERUPRO

1. The user decides which axes/spindles should actually be physically traversed. "Suppress program test" is set for these axes.
2. The "program test" state is selected for channels 1 and 3.
3. Channels 1, 2 and 3 are started via the PLC.
4. A fault or an alarm occurs, the user interrupts with RESET.

5. SERUPRO at the interruption location of all 3 channels.
6. Search destination has been reached in all 3 channels.
7. Start all 3 channels.
8. Channels 1 and 3 are now again in "program test" and "channel-by-channel running-in" is continued.

Example 2: Activating "suppress program test"

A channel is in program test. In operation, "suppress program test" should be initiated for axis "Y" (at block N1010).

Program code	Comment
N1000 G0 Y1000	
N1010 G4 F10	
N1020 G0 G91 Y=10	; Incremental traversing.
N1030 M30	

With this sequence, the program moves to position 1010, i.e. the simulated component "1000" of this axis is moved after activating "suppress program test".

Example 3: Program test and axis interchange

In the following example, axes "X" from channel 1 and "X1" from channel 2 define the 1st axis of the NCK. At the start, all axes are at position 0.

Channel 1 with "Program test"	Channel 2 without "Program test"
N10010 G0 X0 Y0	
N10020 X100	
N10030 WAITM(91,1,2)	
N10040 WAITM(92,1,2)	
N10050 M0	
N10060 M30	
	N20010 WAITM(91,1,2)
	N20020 G91 G0 X1=10
	N20030 WAITM(92,1,2)
	N20040 M0
	N20050 M30

With block N20040 (M0) the 1st axis (X or X1) is physically located at position 110! This means that the simulated position reached in channel 1 of "100" is assumed in channel 2 with block N20020.

References

For information on the program test, see:

Function Manual Basic Functions; K1: Mode Group, Channel, Program Operation, Reset Response

5.4 Axis/spindle replacement

5.4.1 Introduction

General information

An axis/a spindle is permanently assigned to a specific channel via machine data. The axis/spindle can be used in this channel only.

Definition

With the function "Axis or spindle replacement" it is possible to enable an axis or a spindle and to allocate it to another channel, that means to replace the axis/spindle.

Since the spindle function is subordinated to the axis function, only the term "Axis replacement" is used in the following.

Axis types

According to the channel, we distinguish four types of axes: The reactions at axis change depend on the settings in MD 30552:

MD30552 $MA_AUTO_GET_TYPE

Channel axis

A channel axis can be programmed in the part program and traversed in all modes.

PLC axis

A PLC axis can only be positioned via the PLC.

If a PLC axis is programmed in the part program

- in case of MD AUTO_GET_TYPE = 0 an alarm will be output.
- in case of MD AUTO_GET_TYPE = 1 an automatic GET will be generated.
- in case of MD AUTO_GET_TYPE = 2 an automatic GETD will be generated.

Neutral axis

If a neutral axis is programmed in the part program

- in case of MD AUTO_GET_TYPE = 0 an alarm will be output.
- in case of MD AUTO_GET_TYPE = 1 an automatic GET will be generated.
- in case of MD AUTO_GET_TYPE = 2 an automatic GETD will be generated.

Axis in another channel

This is actually not a proper type of axis. It is the internal state of a replaceable axis. If this happens to be active in another channel (as channel, PLC or neutral axis).

If an axis is programmed in another channel in the part program:

- in case of MD AUTO_GET_TYPE = 0 an alarm will be output.
- in case of MD AUTO_GET_TYPE = 1 an automatic GET will be generated.
- in case of MD AUTO_GET_TYPE = 2 an automatic GETD will be generated.

Note

Both machine data:

MD20110 $MC_RESET_MODE_MASK

MD20112 $MC_START_MODE_MASK

control the behavior of axis assignments in RESET, during booting and part program start. The settings for channels between which axes are to be replaced must be selected such that no illegal constellations (alarms) are generated in conjunction with the following machine data:

MD30552 $MA_AUTO_GET_TYPE

References:

Function Manual, Basic Functions; Axes, Coordinate Systems, Frames (K2), Section: Workpiecerelated actualvalue system

---

Requirements

In order for an axis replacement to be executed, the valid channel for the machine axis number must be specified via the **channel-specific** machine data

MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in channel)

and the delete setting of the channel for axis replacement via the **axis-specific** MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN (channel for axis replacement).

This results in the following regulations:

1. In which channel can the axis be used and replaced?
2. To which channel shall the axis be allocated with power ON?

Example of an axis replacement between channels

With 6 axes and 2 channels, the 1st, 2nd, 3rd and 4th axis in channel 1 and the 5th and 6th axis in channel 2 shall be used. It shall be possible to replace the 1st axis, this shall be allocated to channel 2 after power ON.

The **channel-specific** machine data must be allocated with:

CHANDATA(1)
MD20070 $MC_AXCONF_MACHAX_USED=(**1**, 2, 3, 4, 0, 0, 0, 0)

CHANDATA(2)
MD20070 $MC_AXCONF_MACHAX_USED=(5, 6, **1**, 0, 0, 0, 0, 0)

The **axis-specific** machine data must be allocated with:

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN[AX1]=2

Display

The current axis type and the current channel responsible for this axis are displayed in an axial PLC interface byte (see Section "Axis replacement via the PLC").

Note

If an axis is not valid in the selected channel, this is indicated by inverting the axis name on the user interface.

5.4.2 Example of an axis replacement

Assumption

With 6 axes and 2 channels, the 1st, 2nd, 3rd and 4th axis in channel 1 and the 5th and 6th axis in channel 2 shall be used. It shall be possible to replace the 2nd axis between the channels and to allocate to channel 1 after power ON.

Exercise

The task is subdivided into the following areas:

- Machine data allocation so that the prerequisites for axis replacement are given.
- Programming of axis replacement between channel 1 and channel 2.

Fulfillment of preconditions

Population of channel-specific machine data MD20070

$MC_AXCONF_MACHAX_USED[1]=(1, **2**, 3, 4, 0, 0, 0, 0)

$MC_AXCONF_MACHAX_USED[2]=(5, 6, **2**, 0, 0, 0, 0, 0)

Population of the axis-specific machine data:

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN[AX2]=1

Program in channel 1

```
...
RELEASE (AX2)
            ; Release the axis AX2
INIT (2, "_N_MPF_DIR\_N_TAUSH2_MPF", "S")
            ; Select program TAUSH2 in channel 2
START (2)
            ; Start program in channel 2
WAITM(1,1,2)
            ; Wait for wait flag 1 in channels 1 and 2
...         ;
...         ;Rest of program after axis replacement
...         ;
...         ;
M30
```

Program TAUSH2 in channel 2

```
...
WAITM(1,1,2)
            ; Wait for wait flag 1 in channels 1 and 2
GET (AX2)
...         ; Accept the axis AX2
...         ;
...         ;Rest of program after axis replacement
...         ;
RELEASE (AX2)
...         ; Release for additional axis replacement
...         ;
M30
```

5.4.3 Axis replacement options

One or more axes/spindles can be activated for replacement between channels by the part program or by motion-synchronous actions. An axis/spindle replacement can also be requested and released from the PLC via the VDI interface. The axis/spindle must have been released in the current channel and will be taken over by the other channel with the GET command and released with the RELEASE command.

If the specified conditions are met, an axis/spindle replacement is initiated by:

- Programming in part program GET/GETD

  Take over an axis or a spindle from another channel with GET or get it from another channel with GETD. With GETD a corresponding release is not required.
- Automatic by programming of the axis name, if the required conditions have been met by MD30552 $MA_AUTO_GET_TYPE > 0.
- Without preprocessing stop and existing synchronization between preprocessing and main run.
- Through PLC via the VDI interface to the NCK.

When taking over a PLC-controlled axis, the channel behavior triggered by the NC program can be de-coupled via an interface signal. This permits, for example, individual PLC axes to be interpolated independently of the NC program (see also Section "Autonomous singleaxis operations (Page 608)").

Axis replacement extensions

- Setting axis replacement behavior as variable via machine data MD10722 $MN_AXCHANGE_MASK.
- Axis replacement with an axis container rotation with implicit GET/GETD
- Axis replacement without pre-processing stop of axes not involved in the contour.

- Geometry axis with rotated frame and axis replacement in JOG mode via machine data MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED can be activated.
- Axis replacement via synchronized actions GET(axis), RELEASE(axis), AXTOCHAN, $AA_AXCHANGE_TYP(axis).

5.4.4 Replacement behavior NC program

Possible transitions

The following diagram shows which axis replacement possibilities are available.

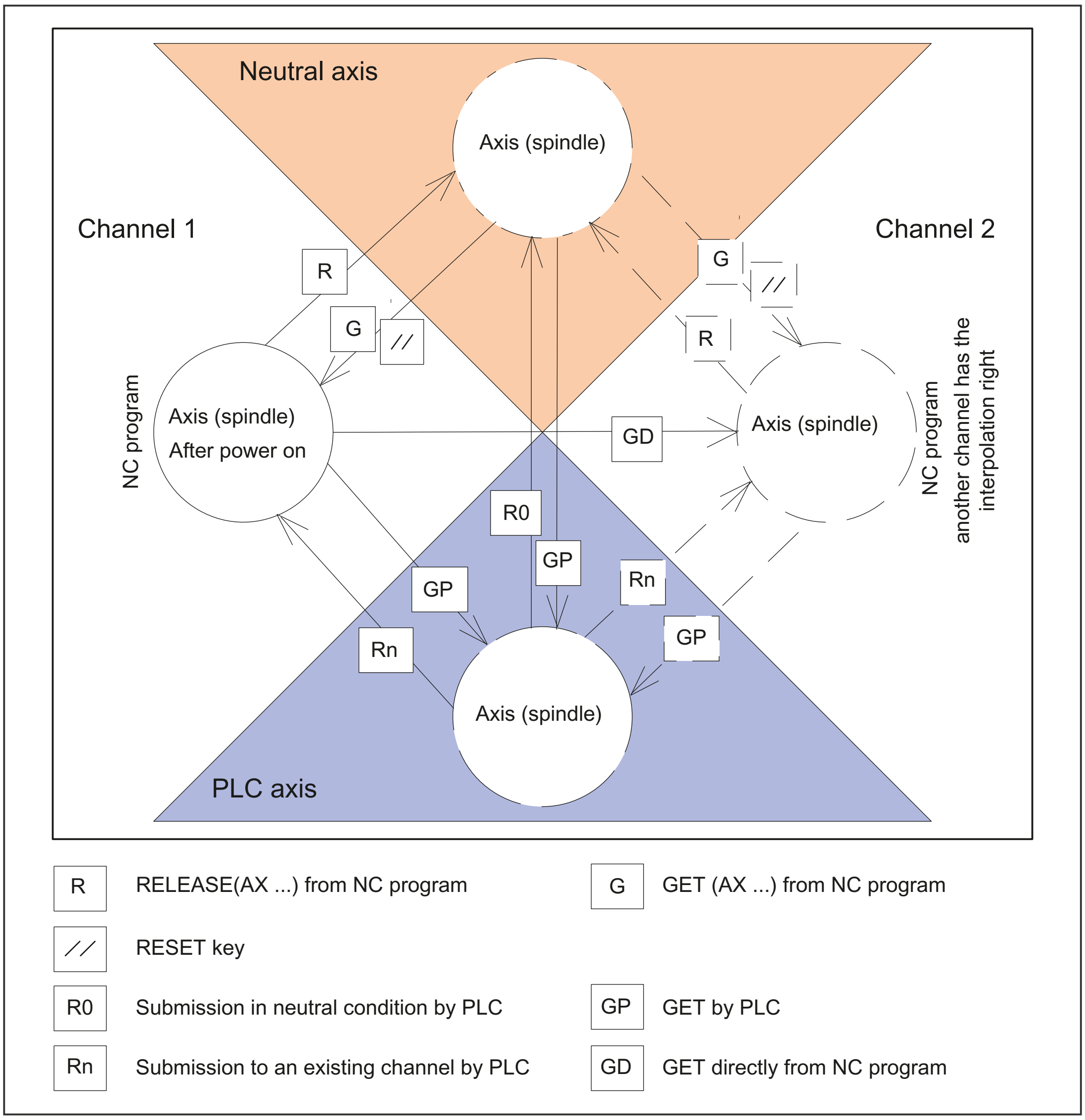


Figure 5-5 Transitions of possible axis states during axis replacement

5.4.5 Transition the axis into the neutral state (RELEASE)

Function

The predefined `RELEASE()` procedure is used to bring an axis/spindle into the "Neutral axis" state.

Syntax

```
RELEASE(<MA>, SPI(<SPNR>)[, ...])
```

Meaning

`RELEASE`:	Axis/spindle brought into the "Neutral axis" state
	Supplementary condition: Must be alone in the block.
`MA`:	Machine axis name
`SPI(...)`:	Function to convert a spindle number into the appropriate axis name
`SPNR`:	Spindle number

More information

General abort conditions

- Not all the requirements for the axis replacement have been met. See machine data
  - MD20070 $MC_AXCONF_MACHAX_USED
  - MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN)
- The axis is involved in a transformation.
- The axis belongs to an axis grouping.

Abort with reset

If a reset is triggered when an axis has been released with `RELEASE()`, but has not yet been taken over by means of `GET()`, the axis is re-assigned to the releasing channel.

Guide axis of a gantry grouping

If the guide axis of a gantry grouping is released, all synchronous axes are also released.

5.4.6 Transferring an axis or spindle into the part program (GET, GETD)

Options

The release time and the behavior of an axis or spindle replacement is influenced in the part program as follows:

- Programming with the GET command in the same channel.
- Directly from another channel through programming with GETD.

References:
Programming Manual Job Preparation; Flexible NC Programming,
Section: Axis replacement, spindle replacement (RELEASE, GET, GETD)

With the GETD command

GET (axis name, axis name, SPI (spindle no.), ...)

The takeover of an axis is delayed if

- the axis is changing the measuring system.
- servo disable is being processed for the axis (transition from control in follow-up/stop and vice versa).
- the axis/spindle disable is set.
- the axis has not yet been enabled by the other channel with RELEASE.
- interpolation for the axis has not yet been completed (except for a speed-controlled spindle).

With GET (axis name, ...) a separate NC block with pre-processing stop is always generated.

Exceptions:

- If the axis is already a channel axis, then no block is generated.
- If the axis is synchronous, (i.e. it has not been swapped to another channel in the meantime or received a signal from the PLC) no extra block is generated either.

With the GETD command

With **GETD** (GET Directly) an axis is fetched directly from another channel. That means that no suitable RELEASE must be programmed for this GETD in another channel. In addition, another channel communication must be created (e.g. wait marks), since the supplying channel is interrupted with GETD. If the axis is a PLC axis, replacement is delayed until the PLC has enabled the axis.

> ⚠ CAUTION
>
> **Function-less axis**
>
> The GETD command interrupts the program run in the channel in which the required axis is currently to be found! (REORG).
>
> Exception: The axis is at the time in a neutral state.

Note

If the GET or GETD command has been programmed, take-over is delayed and the channel is reset; the channel will no longer try to take over the axis.

An axis taken over with GET remains allocated to this channel even after a key RESET or program RESET. The axis can be replaced by programming RELEASE and GET again or will be assigned to the channel defined in the following machine data during POWER ON:

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN

5.4.7 Automatic axis replacement

Automatically through programming of axis name

Depending on machine data MD30552 $MA_AUTO_GET_TYPE a GET or GETD is generated automatically if a neutral axis is again programmed or assigned to another channel.

Preconditions for automatic axis replacement

MD30552 $MA_AUTO_GET_TYPE > 0 must be true for automatic axis replacement.

Automatic GETD

Note

If an automatic GETD is set, the following must be observed:

- Channels could mutually influence each other.
  (REORG, when axis is removed.)
- With simultaneous access of several channels to an axis it is not known which channel will have the axis at the end.

Example 1

Program code	Comment
N1 M3 S1000	
N2 RELEASE (SPI(1))	; => Transition to neutral condition
N3 S3000	; New speed for released spindle
	; MD AUTO_GET_TYPE =
	; 0 =>Alarm "Wrong axis type" is generated
	; 1 => GET (SPI(1)) is generated.
	; 2 => GETD (SPI(1)) is generated.

Example 2

Program code	Comment
	; (axis 1 = X)
N1 RELEASE (AX1	; => Transition to neutral condition
N2 G04 F2	
N3 G0 X100 Y100	; Motion of the released axis
	; MD AUTO_GET_TYPE =
	; 0 =>Alarm "Wrong axis type" is generated
	; 1 => GET (AX1) is generated.
	; 2 => GET (AX1) is generated.

Example 3

Program code	Comment
	; (axis 1 = X)
N1 RELEASE (AX1)	; => Transition to neutral condition
N2 G04 F2	
N3 POS (X) = 100	; Positioning of the released axis
	; MD AUTO_GET_TYPE =
	; 0 =>Alarm "Wrong axis type" is generated
	; 1 => GET (AX1) is generated. *)
	; 2 => GET (AX1) is generated. *)

*) If the axis is still synchronized, no dedicated block will be generated.

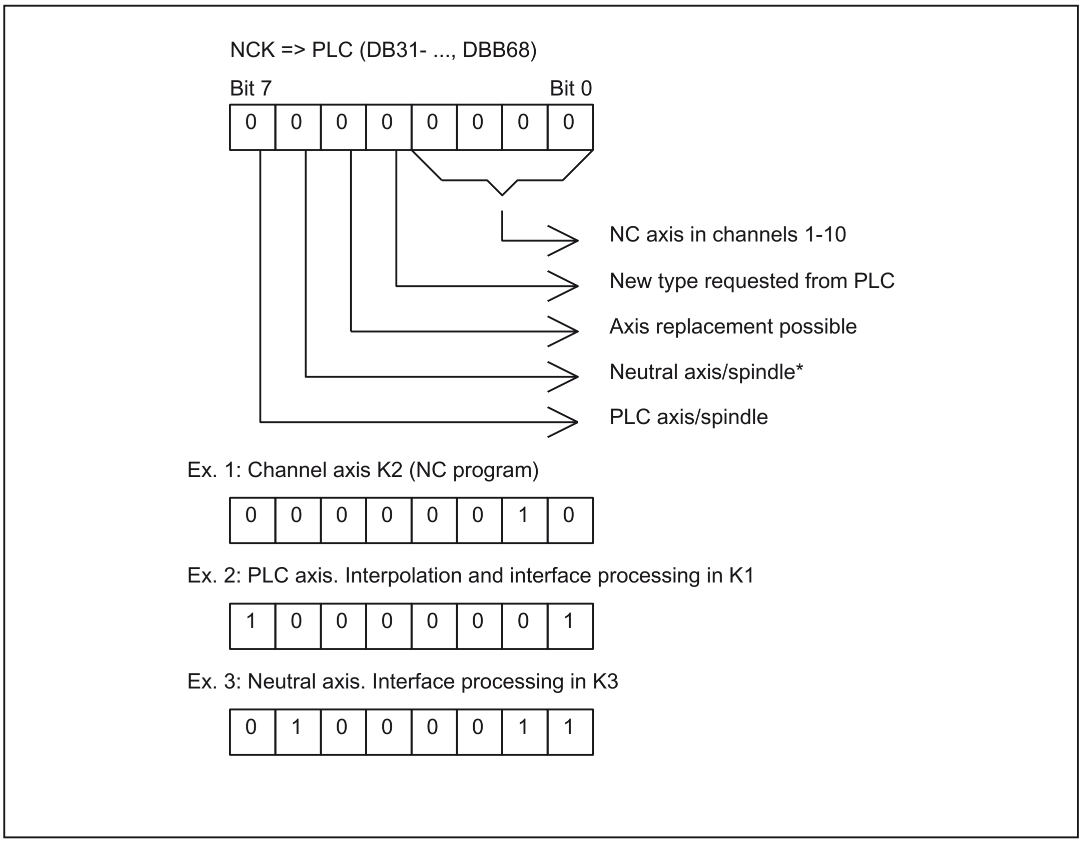
5.4.8 Axis replacement via PLC

The PLC can request and traverse an axis at any time and in any operating mode.

The PLC can change an axis from one channel into the other channel (only for 840D sl).

TYPE display

The type of an axis can be determined at any time via an interface byte (PLC-axis, channel axis, neutral axis).



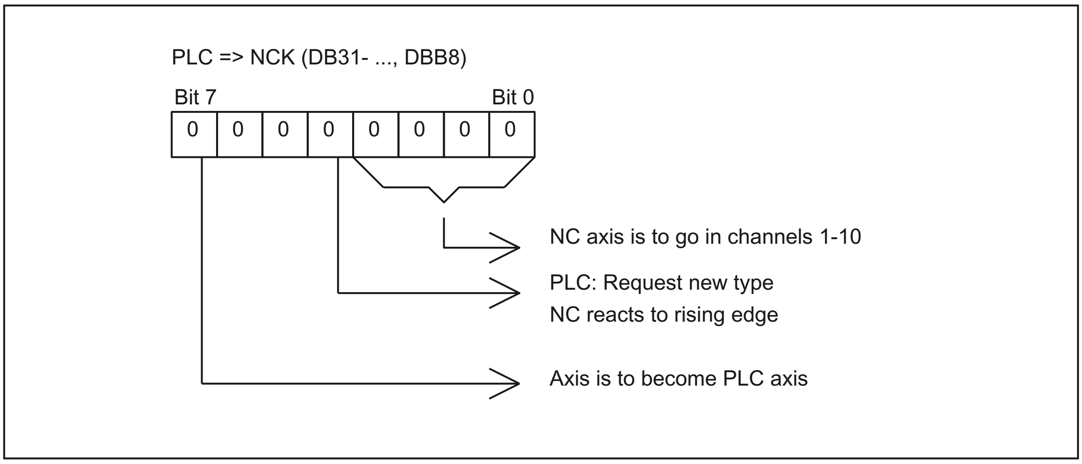
*** neutral axis/spindle** also contains the **Command-/Oscillation-axis**

	NCK=>PLC, DBB68	PLC=>NCK, DBB8
After power ON	0 0 1 0 0 0 0 1	0 0 0 0 0 0 0 0
RELEASE (K1)	0 1 1 0 0 0 0 1	
	0 1 0 0 0 0 0 1	
GET (K2)	0 0 1 0 0 0 1 0	
	0 0 0 0 0 0 1 0	

Time

Figure 5-6 Changing an axis from K1 to K2 via parts program

TYPE input



In principle, the PLC must set the signal "Request new type". It is deleted again after change. Also for a channel interchange with GET and RELEASE (only 840D sl).

Controlling PLC axes/spindles for 840D sl

PLC axes and PLC spindles are traversed using function block FC18 in the basic PLC program

FC18: SpinCtrl Spindle control

Examples

The sequence of NC/PLC interface signals to change an NC axis to a PLC axis and to transition an NC axis into a neutral axis by the PLC are shown in the following diagrams.

	NCK=>PLC, DBB68	PLC=>NCK, DBB8
After power ON	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0
New TYPE (PLC)		1 0 0 1 0 0 0 0
	0 0 1 1 0 0 0 1	1 0 0 1 0 0 0 0
	0 0 1 1 0 0 0 1	1 0 0 0 0 0 0 0
	1 0 0 0 0 0 0 1	1 0 0 0 0 0 0 0

Time

Figure 5-7 Changing an NC axis to a PLC axis

	NCK=>PLC, DBB68	PLC=>NCK, DBB8
After power ON	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0
New TYPE (PLC)		1 0 0 1 0 0 0 0
	0 0 1 1 0 0 0 1	1 0 0 1 0 0 0 0
		1 0 0 0 0 0 0 0
	1 0 0 0 0 0 0 1	1 0 0 0 0 0 0 0
New TYPE (PLC)		0 0 0 1 0 0 0 0
	1 0 1 1 0 0 0 1	0 0 0 1 0 0 0 0
	0 1 0 0 0 0 0 1	0 0 0 0 0 0 0 0

Time

Figure 5-8 Changing an NC axis to a neutral axis through the PLC

5.4.9 Set axis replacement behavior variable.

The axis is replaced in the currently enabled channel and, depending on the respective axis type, the axis replacement behavior can be influenced via machine data MD10722 $MN_AXCHANGE_MASK:

Table 5- 2 Time of release of axes or spindles during replacement

MD10722	Axis replacement behavior
Bit 0 = 1	**Automatic Axis Replacement** between two channels also takes place when the axis has been brought to a neutral state by WAITP.
Bit 1 = 1	Release by **Axis Container Rotation** with implicitly generated GET/GETD When requesting an axis container rotation, all axes of the axis container that can be assigned to the executing channel are taken into the channel by means of an implicit GET or GETD. A subsequent replacement is only allowed after completion of the axis container rotation.
Bit 2 = 1	**Axis replacement without pre-processing** and possible forced re-organization of axes not involved in the contour. When an intermediate block is inserted in the main run, a check will be made to determine whether or not reorganization is required. Reorganization is only necessary if the axis states of this block do **not** match the current axis states.
Bit 3 = 0	PLC replacing of permanently **assigned PLC axis** An axis replacement can be requested from the PLC for each axis. The permanently assigned PLC axis only from neutral axis to PLC axis and vice versa.
Bit 3 = 1	Replacement request via VDI interface A replacement request via VDI interface is only executed for an: **axis exclusively controlled by the PLC** when MD30460 $MA_BASE_FUNCTION_MASK with Bit4=1. **permanently assigned** competing positioning axis (PLC axis) when MD30460 $MA_BASE_FUNCTION_MASK with Bit5=1. For such axes the interface signal NCK → PLC DB31, ... DBX68.5 (replacement possible) is always set to 1. For all other axes this bit is set to 0. For a permanently assigned PLC axis an axis replacement is only possible from neutral axis to PLC axis and vice versa.

5.4.10 Axis interchange via axis container rotation

Enabling axis container rotation

When an axis container rotation is enabled, all container axes that can be allocated to a channel are allocated to this channel using implicitly generated `GET` or `GETD`. An axis can only be relinquished, e.g. to another channel, after container rotation.

Note

The implicit assignment of an axis to a channel is **not**possible if the axis in the state "main run axis" (e.g. is a PLC axis). In order to be able to participate in the axis container rotation, the axis must first exit the state.

For further explanations on the axis replacement of container axes (see Section "B3: Distributed systems - 840D sl only (Page 71)").

Example: Axis container rotation with an implicit `GET` or `GETD`

Action Channel 1	Action Channel 2
	SPOS = 180 positioned
`AXCTSWE(CT 1)`	; gets spindle in Channel 1
	; and allows axis container rotation

Assumption:

The spindle is used in both channels and is also an axis in axis container CT 1.

Activation

Axis interchange using axis container rotation and implicit `GET`/`GETD` is activated using machine data MD10722 $MN_AXCHANGE_MASK, bit 1=1.

5.4.11 Axis replacement with and without preprocessing stop

Axis replacement extension without preprocessing stop

Instead of a GET block with a preprocessing stop, this GET request only generates an intermediate block. In the main run, when this block is executed, the system checks whether the states of the axes in the block match the current axis states. If they do not match, forced reorganization can be triggered.

The following states of an **axis or positioned spindle** are checked for:

- The mode, either for the axis or for positioned spindle
- Setpoint position

The following states of a **Spindle in speed mode** are checked:

- Spindle mode: Speed mode
- Spindle speed S
- Direction of rotation M3, M4
- Gear stage M40, M41, M42, M43, M44, M45
- Master spindle at constant cutting rate.

In some instances, forced reorganization may be possible. Reorganization of the following axes is forced in any case.

Activation

Replacement without preprocessing and checking of the current states is activated with machine data MD10722 $MN_AXCHANGE_MASK, Bit 2=1.

Example

Activating an axis replacement without a preprocessing stop

```
N010 M4 S1000
N011 G4 F2
N020 M5
N021 SPOS=0
N022 POS[B]=1
N023 WAITP(B)                                   ; Axis b becomes the neutral axis
N030 X1 F10
N031 X100 F500
N032 X200
N040 M3 S500
N041 G4 F2
N050 M5
N099 M30
```

If the spindle (axis B) is traversed immediately after block N023 as a PLC axis to 180° and back to 1°, and then again to the neutral axis, block N040 does not trigger a preprocessing stop nor a reorganization.

Special case: Axis replacement with preprocessing stop

Without a GET or GETD instruction having previously occurred in the main run, the spindle or the axis can be made available again by RELEASE (axis) or WAITP (axis), for example. A subsequent GET leads to a GET **with** a preprocessing stop.

5.4.12 Axis exclusively controlled from the PLC

Function

After the control boots, the axis is in the "neutral axis" state. The PLC controls it. To traverse the axis as competing positioning axis (from the PLC via function block FC18), the axis must first be explicitly requested from the PLC.

Note

Per machine data, the axis interchange to the PLC can be exclusively restricted to PLC controlled axes: MD10722 $MN_AXCHANGE_MASK, Bit 3 = 1

The axis **cannot** be traversed from an NC part program.

Parameter assignment

Parameterizing an axis as axis that is exclusively controlled from the PLC is realized using the axis-specific machine data:

MD30460 $MA_BASE_FUNCTION_MASK, Bit 4 = 1

Control by PLC

The traversing behavior of an axis exclusively controlled from the PLC is only influenced by the axial NC/PLC interface signals:

- DB31, ... DBX28.1 (reset)
- DB31, ... DBX28.2 (continue)
- DB31, ... DBX28.6 (stop along braking ramp)

Possible traversing functions

The following traversing functions are possible for axes exclusively controlled from the PLC:

1. Traversing in the JOG mode using the traversing keys and handwheel
2. Referencing the axis
3. Traversing as command axis via static synchronized actions
4. Traversing as asynchronous oscillating axis
5. Traversing as competing positioning axis from the PLC via FC18

After traversing functions 1. to 4. have been completed, the axis automatically goes back into the "neutral axis" state. After traversing function 5. from the PLC has been completed, the axis remains in the state "PLC axis". The axis only changes into the "Neutral axis" state after having been explicitly released by the PLC.

5.4.13 Axis permanently assigned to the PLC

Function

After the control has booted, the axis is in the "neutral axis" state and is controlled from the NC channel. To traverse the axis as competing positioning axis (from the PLC via function block FC18), the axis **does not** have to be explicitly requested from the PLC. Axis interchange to the PLC is realized automatically using the traversing request via FC18. After the traversing motion requested via FC18 has been completed, the axis again automatically changes into the "neutral axis" state.

After the axis has been interchanged, and after the request from the PLC, the axis can also be controlled from the PLC: "PLC axis" state.

Note

Per machine data, the axis interchange to the PLC can be exclusively restricted to axes that are permanently assigned to the PLC: MD10722 $MN_AXCHANGE_MASK, Bit 3 = 1

Parameter assignment

Parameterizing an axis as axis that is permanently assigned to the PLC is realized using the axis-specific machine data:

MD30460 $MA_BASE_FUNCTION_MASK, Bit 5 = 1

Control by the PLC or NC channel

The traversing behavior of an axis permanently assigned to the PLC can either be influenced by the NC channel or by the PLC:

NC channel: Channel-specific NC/PLC interface signals (selection)

- DB21, ... DBXDBX7.1 (NC start)
- DB21, ... DBXDBX7.3 (NC stop)
- DB21, ... DBXDBX7.7 (reset)

PLC: Axial NC/PLC interface signals

- DB31, ... DBX28.1 (reset)
- DB31, ... DBX28.2 (continue)
- DB31, ... DBX28.6 (stop along braking ramp)

Possible traversing functions

The following traversing functions are possible for an axis permanently assigned to the PLC:

1. Traversing in the JOG mode using the traversing keys and handwheel
2. Referencing the axis
3. Traversing as competing positioning axis from the PLC via FC18

After traversing functions 1. to 3. have been completed, the axis automatically goes back into the "neutral axis" state.

5.4.14 Geometry axis in rotated frame and axis replacement

Replacement expansion via Frame with Rotation

In JOG mode, a geometry axis with rotated frame can be traversed as PLC axis or as a command axis via static synchronized actions. In order to achieve this, in machine data MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED, bit 10=1 must be set. The reposition behavior of this axis is influenced via Bit 11.

Note

Before changing operational mode during JOG mode

Before changing the operational mode from JOG mode, all traverse motions of **all** PLC and command axes, which have been linked as geometry axes in the rotated frame, must have been concluded. These axes must at least have become neutral axes again, otherwise alarm 16908 will be generated when the operational mode is changed. This alarm is also generated when only a single geometry axis is traversed as a PLC or command axis in the rotated coordinate system.

Such an axis can only become a PLC or command axis within the channel, an axis replacement in another channel is not allowed.

Prerequisite for changing from JOG to AUTOMATIC

When changing from JOG mode to AUTOMATIC, the Condition program is interrupted and the end point of this geometry axis motion is only taken over if in MD 32074: FRAME_OR_CORRPOS_NOTALLOWED bit 11=1. This positions the PLC or command axes in relation to the rotation of the frame.

All axes influenced by a rotating frame are considered as geometry axes grouping and are handled collectively. In this way, all axes of the

- assigned to the NC program or
- all axes are neutral or
- are active as main run axes (PLC, command, or oscillation axis).

For example, if one axis is programmed with a WAITP, waiting is performed for all further axes of the geometry axis grouping, so that these axes can collectively become neutral axes. If one of the axes becomes a PLC axis in the main run, then all other axes of this grouping become neutral axes.

Supplementary conditions

If MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED, bit 10 == 0 and `ROT Z45` is programmed in the NC program, then for the X and Y axes **no axis interchange** is possible. This is also analogously valid for the Z axis for e.g. `ROT X45` or `ROT Y45` – and also in the JOG operating mode – if a block was interrupted with this type of programming. Although in this case the NC/PLC interface signals are set for the X and Y axes:

- DB31, ...DBX68.5 (axis interchange possible) = 1
- DB32, ...DBX68.5 (axis interchange possible) = 1

However, these are reset.

Only if MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED, Bit 10 == 1 and no block with this programming is being currently traversed, then in the JOG mode, these types of axes can be interchanged.

5.4.15 Axis replacement from synchronized actions

Function

An axis can be requested with GET(axis) and be released for axis replacement with RELEASE(axis) with a synchronous action.

Note

The axis must be assigned as a channel axis via machine data.

An axis can be transferred directly between channels to a certain channel with the NC language command AXTOCHAN via synchronized actions or in the part program. This axis does not have to be the same channel and it is not necessary that this channel be in possession of the current interpolation right for the axis.

Current state and interpolation right of the axis

With which axis type and interpolation right a possible axis replacement is to be performed, can be deducted from the system variable $AA_AXCHANGE_TYP[axis].

0: The axis is assigned to the NC program

1: Axis assigned to PLC or active as command axis or oscillating axis

2: Another channel has the interpolation right.

3: Axis is neutral axis.

4: Neutral axis is controlled by the PLC.

5: Another channel has the interpolation right, axis is requested for NC program.

6: Another channel has the interpolation right, axis is requested as neutral axis.

7: Axis active for PLC or as command or oscillating axis, axis is
requested for PLC program.

8: Axis active for PLC or as command or oscillating axis, axis is
requested as neutral axis.

9: Permanently assigned PLC axis, in state of neutral axis.

10: Permanently assigned PLC axis, controlled by PLC, in state of neutral axis.

Permanently assigned PLC axis

in state of neutral axis $AA_AXCHANGE_TYP = 9 and
controlled by PLC, in state of neutral axis $AA_AXCHANGE_TYP = 10

will be assigned to PLC **independently of GET and RELEASE** permanently.

Whether the axis can also be replaced is displayed via the system variable $AA_AXCHANGE_STAT[axis].

State transitions GET, RELEASE from synchronous actions and when GET is completed

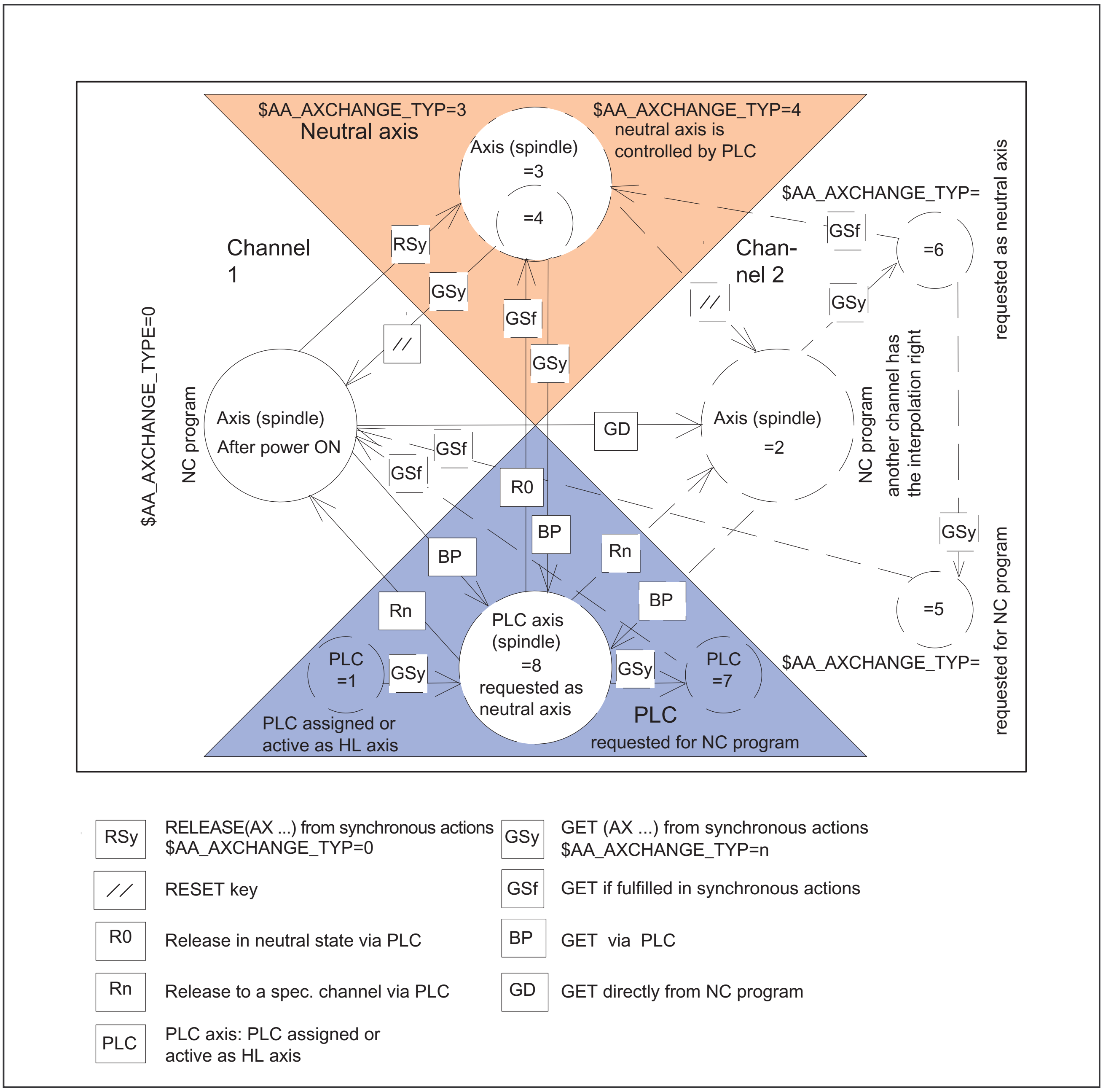


Figure 5-9 Transitions from synchronized actions

For more information, please refer to:

References:

Function Manual, Synchronized Actions; Section: Actions in synchronized actions

5.4.16 Axis interchange for leading axes (gantry)

Function

A closed gantry grouping is treated regarding its axes always as a unit regarding axis interchange. This is the reason why for an axis interchange of the leading axis, an axis interchange is simultaneously made for all synchronous axes of the gantry grouping. In addition to the preconditions for the leading axis described in the previous chapters, the appropriate preconditions must also be fulfilled for all synchronous axes of the gantry grouping.

Axial machine data

For an axis interchange, the following axial machine data must be set the same for all axes of a closed gantry group:

- MD30460 $MA_BASE_FUNCTION_MASK, Bit 4 (control executing components)
- MD30460 $MA_BASE_FUNCTION_MASK, Bit 5 (assignment to components)

Axial NC/PLC interface signals

Within the scope of the axis interchange function, the following axial NC/PLC interface signals always have the same values for all axes of a closed gantry grouping:

- DB31, ... DBX63.0 (reset executed)
- DB31, ... DBX63.1 (PLC controlled axis)
- DB31, ... DBX63.2 (axis stop active)

Axial system variable

Within the scope of the axis interchange function, the following axial system variables always have the same values for all axes of a closed gantry group:

- $AA_AXCHANGE_TYP (axis type regarding axis interchange)
- $AA_AXCHANGE_STAT (axis status regarding axis interchange)
- $AA_SNGLAX_STAT (axis type of the individual axis)

5.5 Marginal conditions

Mode group

Up to 10 mode groups are available for SINUMERIK 840D sl.

Only 1 mode group is available for SINUMERIK 828D.

Channels

Up to 10 channels are available for SINUMERIK 840D sl.

Only 1 channel is available for SINUMERIK 828D.

Axis/spindle interchange

For SINUMERIK 828D, an axis/spindle interchange is not possible between channels.

Change to the channel axis

If an axis is changed from PLC axis, neutral axis or axis in another channel to the axis type channel axis, a synchronization must take place.

With this synchronization,

- the current positions are assumed
- the current speed and gear stage is assumed with spindles.

It is therefore obligatory to perform a feed stop which interrupts the active path movement.

If the axis is transferred with GET, this transition is clearly defined by the parts program.

If the axis is allocated by the PLC, the program section in which the change takes place is not clearly foreseeable.

(Except by a separate user-specific NC <-> PLC logic)

For this reason, the change to the channel axis is delayed in the following conditions:

- Path operation is active (G64+ axes programmed)
- Thread cutting/tapping is active (G33/G331/G332)

Change from a channel axis

The change of a channel axis to a neutral axis or PLC axes cannot be performed during an active path operation.

With RELEASE this is caused by the fact that RELEASE must be located in a separate NC block.

If the PLC changes the axis type, a REORG is triggered internally. Therefore, the change with the listed program conditions is delayed.

Block search

During block search with calculation, all GET, GETD or RELEASE blocks are stored and output after the next NC Start.

Exception:

Blocks which are mutually exclusive are deleted.

Example:

N10	RELEASE (AX1)	Blocks are deleted.
N40	GET (AX1)	"
N70	Target	

5.6 Data lists

5.6.1 Machine data

5.6.1.1 General machine data

Number	Identifier: $MN_	Description
10010	ASSIGN_CHAN_TO_MODE_GROUP[n]	Channel valid in mode group [Channel No.]: 0, 1
10722	AXCHANGE_MASK	Parameterization of the axis replacement response

5.6.1.2 Channel-specific machine data

Basic machine data of channel

Number	Identifier: $MC_	Description
20000	CHAN_NAME	Channel name
20050	AXCONF_GEOAX_ASSIGN_TAB[n]	Assignment of geometry axis to channel axis [GEOAxisNo.]: 0...2
20060	AXCONF_GEOAX_NAME_TAB[n]	Geometry axis name in channel [GEOAxisNo.]: 0...2
20070	AXCONF_MACHAX_USED[n]	Machine axis number valid in channel [Channel axis No.]: 0...7
20080	AXCONF_CHANAX_NAME_TAB[n]	Name of channel axis in the channel [Channel axis No.]: 0...7
20090	SPIND_DEF_MASTER_SPIND	Initial setting of master spindle in channel
20100	DIAMETER_AX_DEF	Geometry axis with transverse axis function
20110	RESET_MODE_MASK	Determination of basic control settings after Reset/TP End
20112	START_MODE_MASK	Determination of basic control settings after NC start
20150	GCODE_RESET_VALUES[n]	Reset G groups [G-Group No.]: 0...59
20160	CUBIC_SPLINE_BLOCKS	Number of blocks for C spline
20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of NC block for compression
20200	CHFRND_MAXNUM_DUMMY_BLOCKS	Empty blocks with phase/radii
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Maximum value with DISC

Number	Identifier: $MC_	Description
20230	CUTCOM_CURVE_INSERT_LIMIT	Maximum angle 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 with TRC
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20400	LOOKAH_USE_VELO_NEXT_BLOCK	Look Ahead to programmed following block velocity
20430	LOOKAH_NUM_OVR_POINTS	Number of override switch points for Look Ahead
20440	LOOKAH_OVR_POINTS[n]	Override switch points for LookAhead [Switch point No.]: 0...1
20500	CONST_VELO_MIN_TIME	Minimum time with constant velocity
20600	MAX_PATH_JERK	Pathrelated maximum jerk
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements
20650	THREAD_START_IS_HARD	Acceleration behavior of axis with thread cutting
20700	REFP_NC_START_LOCK	NC start disable without reference point
20750	ALLOW_GO_IN_G96	G0 logic in G96
20800	SPF_END_TO_VDI	Subprogram end to PLC
21000	CIRCLE_ERROR_CONST	Circle end point monitoring constant
21010	CIRCLE_ERROR_FACTOR	Circle end point monitoring factor
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21110	X_AXIS_IN_OLD_X_Z_PLANE	Coordinate system for automatic Frame definition
21200	LIFTFAST_DIST	Traversing path for fast retraction from the contour
21250	START_INDEX_R_PARAM	Number of first channelspecific R parameter

Auxiliary function settings of the channel

Number	Identifier: $MC_	Description
22000	AUXFU_ASSIGN_GROUP[n]	Auxiliary function group [aux. func. no. in channel]: 0...49
22010	AUXFU_ASSIGN_TYPE[n]	Auxiliary function type [aux. func. no. in channel]: 0...49
22020	AUXFU_ASSIGN_EXTENSION[n]	Auxiliary function extension [aux. func. no. in channel]: 0...49
22030	AUXFU_ASSIGN_VALUE[n]	Auxiliary function value [aux. func. no. in channel]: 0...49
22200	AUXFU_M_SYNC_TYPE	Output timing of M functions
22210	AUXFU_S_SYNC_TYPE	Output timing of S functions
22220	AUXFU_T_SYNC_TYPE	Output timing of T functions
22230	AUXFU_H_SYNC_TYPE	Output timing of H functions

Number	Identifier: $MC_	Description
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions
22250	AUXFU_D_SYNC_TYPE	Output timing of D functions
22260	AUXFU_E_SYNC_TYPE (available soon)	Output timing of E functions
22400	S_VALUES_ACTIVE_AFTER_RESET	S function active after RESET
22410	F_VALUES_ACTIVE_AFTER_RESET	F function active after reset
22500	GCODE_OUTPUT_TO_PLC	G functions to PLC
22550	TOOL_CHANGE_MODE	New tool offset for M function
22560	TOOL_CHANGE_M_CODE	M function for tool change

Channel-specific memory settings

Number	Identifier: $MC_	Description
25000	REORG_LOG_LIMIT	Percentage of IPO buffer for log file enable
28000	MM_REORG_LOG_FILE_MEM	Memory size for REORG (DRAM)
28010	MM_NUM_REORG_LUD_MODULES	Number of blocks for local user variables for REORG (DRAM)
28020	MM_NUM_LUD_NAMES_TOTAL	Number of local user variables (DRAM)
28030	MM_NUM_LUD_NAMES_PER_PROG	Number of local user variables per program (DRAM)
28040	MM_LUD_VALUES_MEM	Memory size for local user variables (DRAM)
28050	MM_NUM_R_PARAM	Number of channelspecific R parameters (SRAM)
28060	MM_IPO_BUFFER_SIZE	Number of NC blocks in IPO buffer (DRAM)
28070	MM_NUM_BLOCKS_IN_PREP	Number of blocks for block preparation (DRAM)
28080	MM_NUM_USER_FRAMES	Number of settable Frames (SRAM)
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of block elements for compile cycles (DRAM)
28100	MM_NUM_CC_BLOCK_USER_MEM	Size of block memory for compile cycles (DRAM)
28500	MM_PREP_TASK_STACK_SIZE	Stack size of preparation task (DRAM)
28510	MM_IPO_TASK_STACK_SIZE	Stack size of IPO task (DRAM)

5.6.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30460	BASE_FUNCTION_MASK	Axis functions
30550	AXCONF_ASSIGN_MASTER_CHAN	Reset position of channel for axis change
30552	AUTO_GET_TYPE	Definition of automatic GET
30600	FIX_POINT_POS	Fixed value positions of axes with G75
32074	FRAME_OR_CORRPOS_NOTALLOWED	Frame or HL offset are not allowed
33100	COMPRESS_POS_TOL	Maximum deviation with compensation

5.6.2 Setting data

5.6.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42000	THREAD_START_ANGLE	Start angle for thread
42100	DRY_RUN_FEED	Dry run feedrate

5.6.3 Signals

5.6.3.1 Signals to/from BAG

The mode group signals from the PLC to the NCK and from the NCK to the PLC are included in data block 11.

The signals are described in:
Reference:
Function Manual, Basic Functions; NC/PLC Interface Signals (Z1),
Chapter "Mode group, Program Operation (K1)"

5.6.3.2 Signals to/from Channel

The channel signals from the PLC to the NCK and from the NCK to the PLC are included in data blocks 21, 22, ... for the first, second ... channel.

The signals are described in:
Reference:
Function Manual, Basic Functions; NC/PLC Interface Signals (Z1),
Chapter "Mode group, Program Operation (K1)"

M1: Kinematic transformation 6

6.1 Brief description

6.1.1 TRANSMIT (option)

Note

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

The "TRANSMIT" function permits the following services:

- Face-end machining on turned parts in the turning clamp
  - Holes
  - Contours
- A cartesian coordinate system can be used to program these machining operations.
- The control maps the programmed traversing movements of the Cartesian coordinate system onto the traversing movements of the real machine axes (standard situation):
  - Rotary axis (1)
  - Infeed axis vertical to the axis of rotation (2)
  - Longitudinal axis parallel to the axis of rotation (3)

    Linear axes (2) and (3) are perpendicular to one another.
- A tool center offset relative to the turning center is permitted.
- The velocity control makes allowance for the limits defined for rotary motion.
- A path in the cartesian coordinate system must not pass through the turning center point (this restriction applies to SW 2 and 3).

Other system variables

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

6.1.2 TRACYL (option)

Note

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

The function "Cylinder surface transformation (TRACYL)" permits the following services:

Machining of

- Longitudinal grooves on cylindrical bodies,
- Transverse grooves on cylindrical bodies
- Arbitrary groove patterns on cylindrical objects.

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

For machining, lathes with

- X-C-Z kinematics and
- X-Y-Z-C kinematics

are supported..

- The control transforms the programmed traversing movements of the cylinder coordinate system into the traversing movements of the real machine axes (standard applications X-C-Z kinematics TRAFO_TYPE_n = 512):
  - Rotary axis (1)
  - Infeed axis vertical to the axis of rotation (2)
  - Longitudinal axis parallel to the axis of rotation (3)

  Note

  Linear axes (2) and (3) are perpendicular to one another. The infeed axis (2) intersects the rotary axis. This constellation does not permit groove side offset.

- For groove side offset, X-Y-Z-C kinematics is required with the following axes (TRAFO_TYPE_n = 513):
  - Rotary axis (1)
  - Infeed axis vertical to the axis of rotation (2)
  - Longitudinal axis parallel to the axis of rotation (3)
  - Longitudinal axis (4) to supplement (2) and (3) to obtain a right-hand cartesian coordinate system.

  Note

  Linear axes (2), (3) and (4) are perpendicular to one another. This constellation permits groove wall corrections.

- The velocity control makes allowance for the limits defined for rotary motion.

`TRACYL`transformation, without groove side compensation, with additional longitudinal axis (cylinder surface curve transformation without groove side offset TRAFO_TYPE_n= 514)

- Transformation without groove side offset requires only a rotary axis and a linear axis positioned perpendicular to the rotary axis.
- If a machine provides an additional linear axis positioned perpendicular to the rotary axis and first linear axis, this can be utilized to improve tool offset.

6.1.3 TRAANG (option)

Note

The "Inclined axis" option under license is required for the function "Inclined axis (TRAANG)".

The function "Inclined axis (TRAANG)" is intended for grinding applications. It allows the following:

- Machining with inclined infeed axis.
- A cartesian coordinate system can be used for programming purposes.
- The control maps the programmed traversing movements of the Cartesian coordinate system onto the traversing movements of the real machine axes (standard situation): Inclined infeed axis.

6.1.4 Chained transformations

Introduction

Two transformations can be chained so that the motion components for the axes from the first transformation are used as input data for the chained second transformation. The motion parts from the second transformation act on the machine axes.

Chaining options

- The chain may include **two** transformations.
- The **second** transformation must be **"Inclined axis"** (TRAANG).
- The first transformation can be:
  - Orientation transformations (TRAORI), incl. universal milling head
  - TRANSMIT
  - TRACYL
  - TRAANG

For more information about the other transformations, please refer to:
References:
Function Manual, Special Function; 3- to 5-axis Transformations (F2).

6.1.5 Activating transformation machine data via parts program/softkey

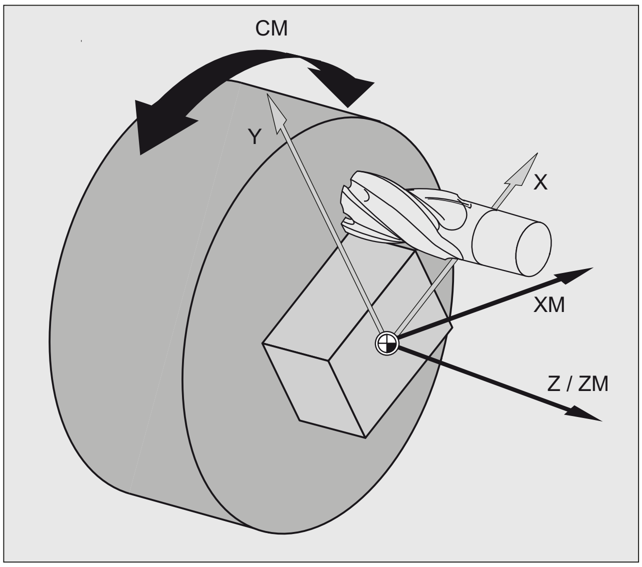
Most of the machine data relevant to kinematic transformations were activated by POWER ON up to now.

Transformation machine data can also be activated via the parts program/softkey and it is not necessary to boot the control.

6.2 TRANSMIT face end transformation (option)

Key statement

The TRANSMIT transformation permits the face end machining on turning machines.



X	Geometry axis
Y	Geometry axis
Z	Geometry axis
XM	Machine axis
ZM	Machine axis
CM	Machine axis

Note

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

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

Machine data

General transformation data

- $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>
- $MC_TRAFO_TYPE_<n>
- $MC_TRAFO_AXES_IN_<n>

with n = 1, 2, 3, ... max. number of transformation data records

---

Note

Maximum of two TRANSMIT transformations per channel

A maximum of two TRANSMIT transformations may be parameterized per channel.

$MC_TRAFO_TYPE_<n> = 256 or 257

TRANSMIT-specific machine data

- $MC_TRANSMIT_ROT_AX_OFFSET_<t>
- $MC_TRANSMIT_ROT_SIGN_IS_PLUS_<t>
- $MC_TRANSMIT_BASE_TOOL_<t>
- $MC_TRANSMIT_POLE_SIDE_FIX_<t>

with t = 1, 2

6.2.1 Specific settings

One rotary and one linear axis: TRAFO_TYPE = 256

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

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

Transformation input axes: $MC_TRAFO_AXES_IN_<n>

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

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

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

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

One rotary and two linear axes: TRAFO_TYPE = 257

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

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

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

Transformation input axes: $MC_TRAFO_AXES_IN_<n>

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

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

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

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

Rotary axis offset: TRANSMIT_ROT_AX_OFFSET

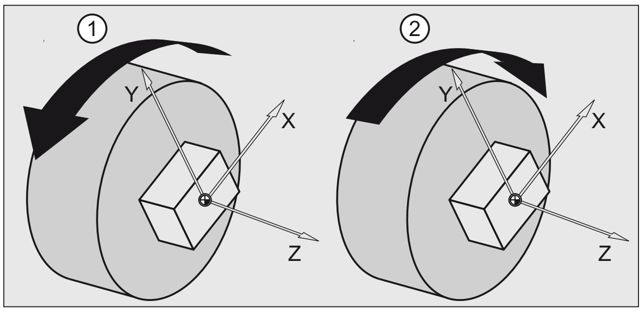
If the rotary axis zero point does not match the rotary axis zero position when the TRANSMIT transformation is active, the angular difference must be entered as an offset in the machine data:

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

Direction of rotation: TRANSMIT_ROT_SIGN_IS_PLUS

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

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



① Positive direction of rotation
② Negative direction of rotation

Figure 6-1 Rotary axis direction of rotation

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

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

Position of tool zero: TRANSMIT_BASE_TOOL

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

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

with t = 1, 2

Replaceable geometry axes

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

- MD22534 $MC_TRAFO_CHANGE_M_CODE = <M function>

Note

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

References:

Function Manual Basic Functions; K2, "Coordinate Systems, Axis Types, Axis Configurations, Workpiece-related Actual-Value System, External Zero Offset"

6.2.2 Switch on

The TRANSMIT transformation is activated in the part program or synchronous action using the `TRANSMIT` command.

Syntax

```
TRANSMIT
TRANSMIT(n)
TRAFOOF
```

Meaning

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

6.2.3 Deactivation

A TRANSMIT transformation active in the channel is activated with:

- Deactivate transformation: `TRAFOOF`
- Activate another transformation, e.g. `TRACYL`, `TRAANG`, `TRAORI`

6.2.4 Applications

Introduction

The transformation TRANSMIT has a pole at the zero point of the TRANSMIT plane (example, see figure: 2–1, x = 0, Y = 0). The pole is located on the intersection between the radial linear axis and the rotary axis (X and CM). In the vicinity of the pole, small positional changes in the geometry axes generally result in large changes in position in the machine rotary axis. The only exceptions are linear motions into or through the pole.

A tool center point path through the pole does not cause the parts program to be aborted. There are no restrictions with respect to programmable traversing commands or active tool radius compensations. Nevertheless, workpiece machining operations close to the pole are not recommended since these may require sharp feedrate reductions to prevent overloading of the rotary axis.

New features

Definition:

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

The following cases are covered:

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

Pole traversal

The pole can be traversed by two methods:

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

Traversal along linear axis

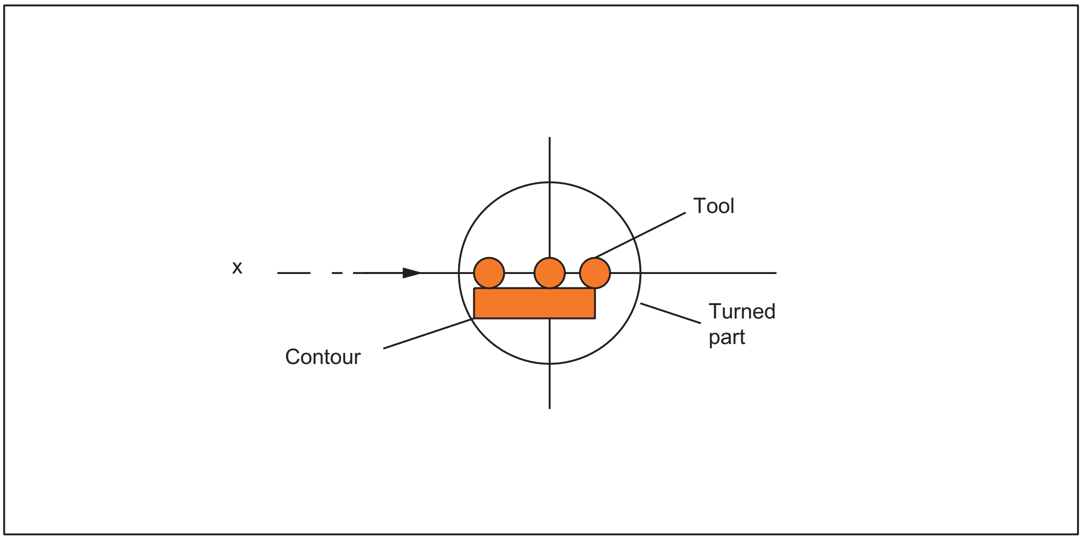


Figure 6-2 Traversal of x axis through pole

Rotation in pole

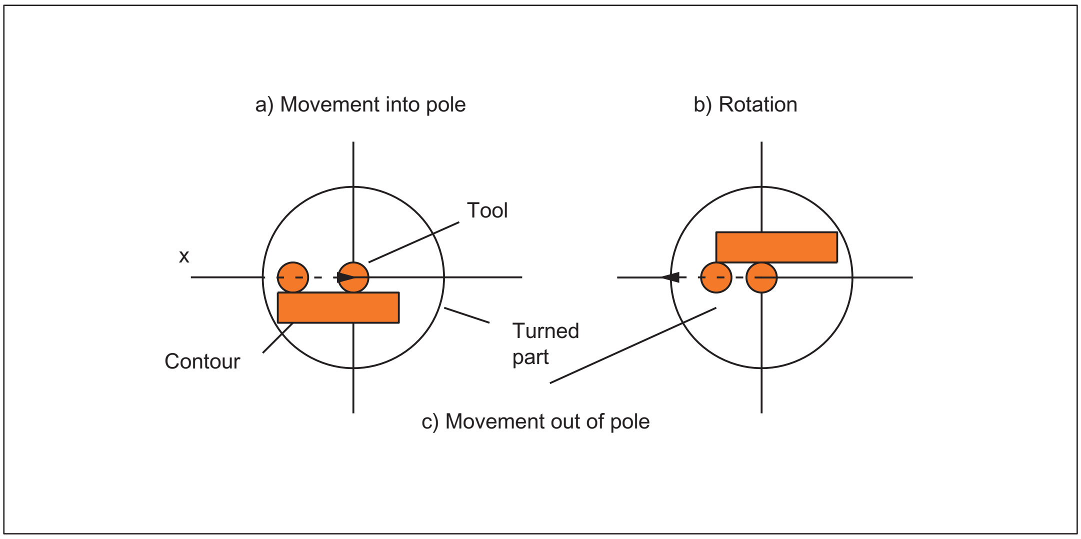


Figure 6-3 Traversal of x axis into pole (a), rotation (b), exit from pole (c)

Selection of method

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

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2

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

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

Special features relating to pole traversal

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

System response:

Table 6- 1 Traversal of pole along the linear axis

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

Special features relating to rotation in pole

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

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 or 2

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 1 or 2

Value: 1 Linear axis remains within positive traversing range

Value: 2 Linear axis remains within negative traversing range

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

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

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

Traversal close to pole

If a tool center point traverses past the pole, the control system automatically reduces the feedrate and path acceleration rate such that the settings of the machine axes (MD 32000 $MA_MAX_AX_VELO[AX*] and MD32300 $MA_ MAX_AX_ACCEL[AX*]) are not exceeded. The closer the path is to the pole, the greater the reduction in the feedrate.

Tool center point path with corner in pole

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

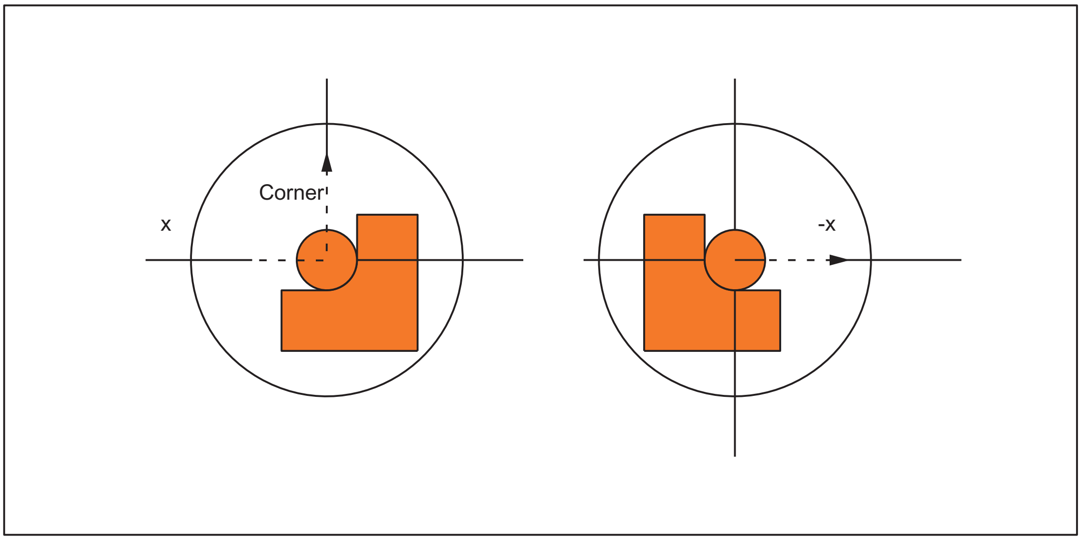


Figure 6-4 Pole traversal

Requirements:

AUTOMATIC mode,

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 0

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 0

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

Corner without pole traversal

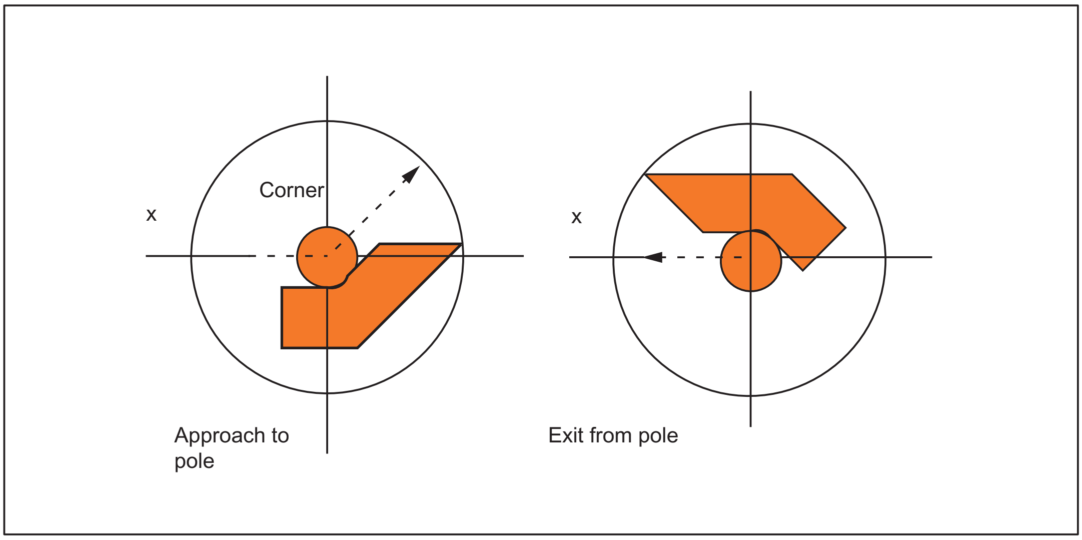


Figure 6-5 Machining on one pole side

Requirements:

AUTOMATIC mode,

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 or 2

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 1 or 2

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

Transformation selection in pole

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

If

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 0

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 0

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

For

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 1

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

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 2

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 2

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

Transformation selection outside pole

The control system moves the axes involved in the transformation without evaluating machine data MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_t. In this case, t = 1 stands for the first and t = 2 for the second TRANSMIT transformation in the channel.

6.2.5 Working area limitations

Starting position

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

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

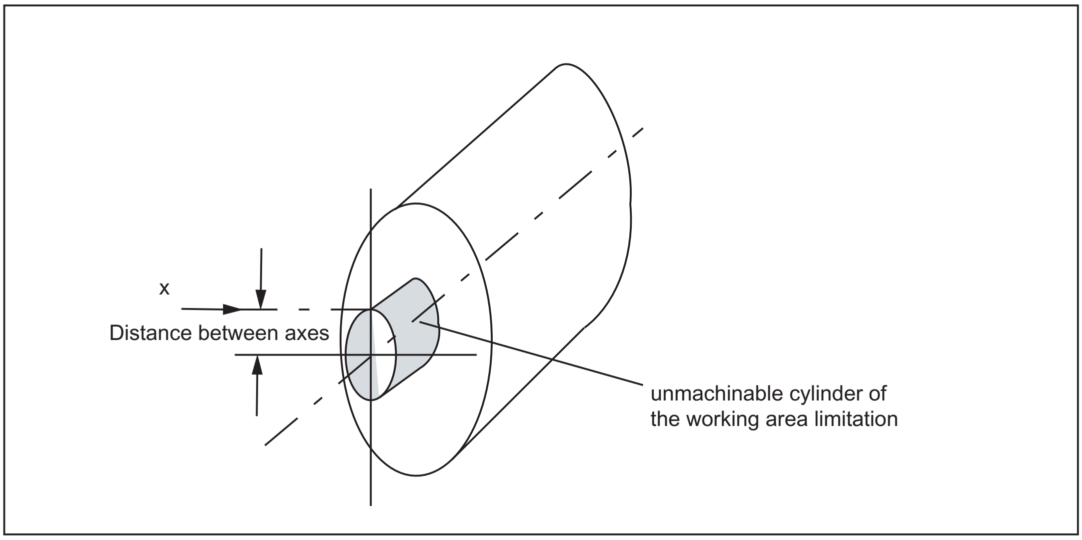


Figure 6-6 Working area limitation based on offset linear axis

Traverse into working area limitation

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

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

Response close to working area limitation

If a tool center point traverses past the prohibited range, the control system automatically reduces the feedrate and path acceleration rate such that the settings of the machine axes (MD 32000 $MA_MAX_AX_VELO[AX*] and MD32300 $MA_ MAX_AX_ACCEL[AX*]) are not exceeded. The closer the path is to the working area limitation, the greater the reduction in the feedrate may be.

6.2.6 Overlaid motions with TRANSMIT

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

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

6.2.7 Monitoring of rotary axis rotations over 360º

Ambiguity of rotary axis positions

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

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

Reference:
Function Manual Basic Functions; "H2: Auxiliary function outputs to the PLC"
Function Manual Synchronized Actions

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

6.2.8 Constraints

Look Ahead

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

Selection of method

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

Several pole traversals

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

Rotary axis as modulo axis

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

Rotary axis as spindle

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

TRANSMIT with supplementary linear axis

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

REPOS

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

Block search

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

6.2.9 Example: Axis configuration

Example of a TRANSMIT axis configuration with two linear axes and one rotary axis.

Channel axes

Geometry axis names

- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1]="Y"
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"

Assignment of geometry axes to channel axes

- TRANSMIT not active
  - MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1 (1st geometry axis ⇒ 1st channel axis "XC")
  - MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0 (-)
  - MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 2 (3rd geometry axis ⇒ 2nd channel axis "ZC")
- TRANSMIT active
  - MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 1 (1st geometry axis ⇒ 1st channel axis "XC")
  - MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 3 (2nd geometry axis ⇒ 3rd channel axis "CC")
  - MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 2 (3rd geometry axis ⇒ 2nd channel axis "ZC")

Channel axis names

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

Assignment of channel axes to machine axes

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

Machine axes

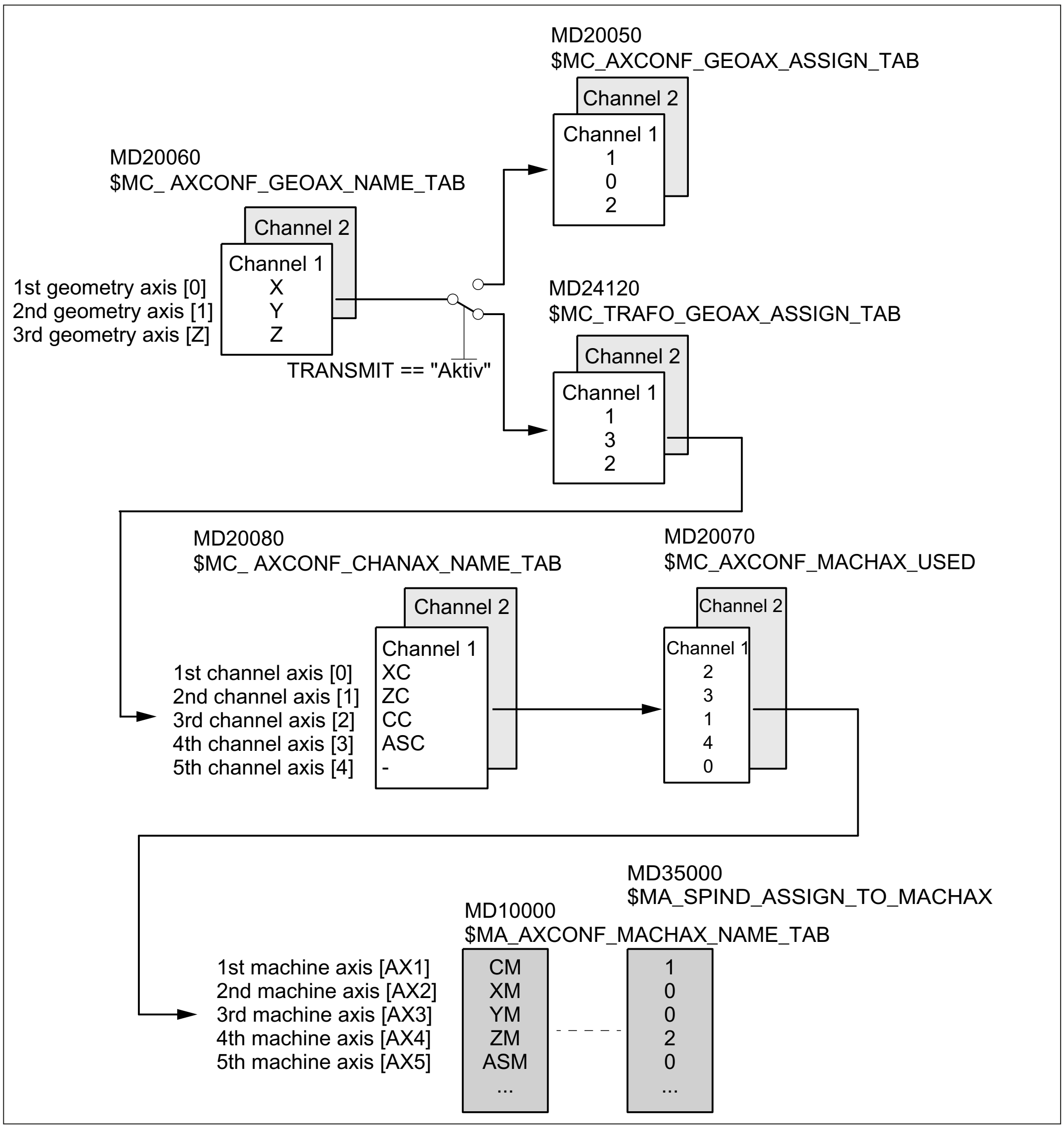
Spindles

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

Machine axis name

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

Synopsis of the axis configuration



6.3 TRACYL cylinder surface transformation (option)

The TRACYL transformation permits the machining of cylinder jacket curves (grooves) on turning machines. The machine kinematics must correspond to the cylinder coordinate system:

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

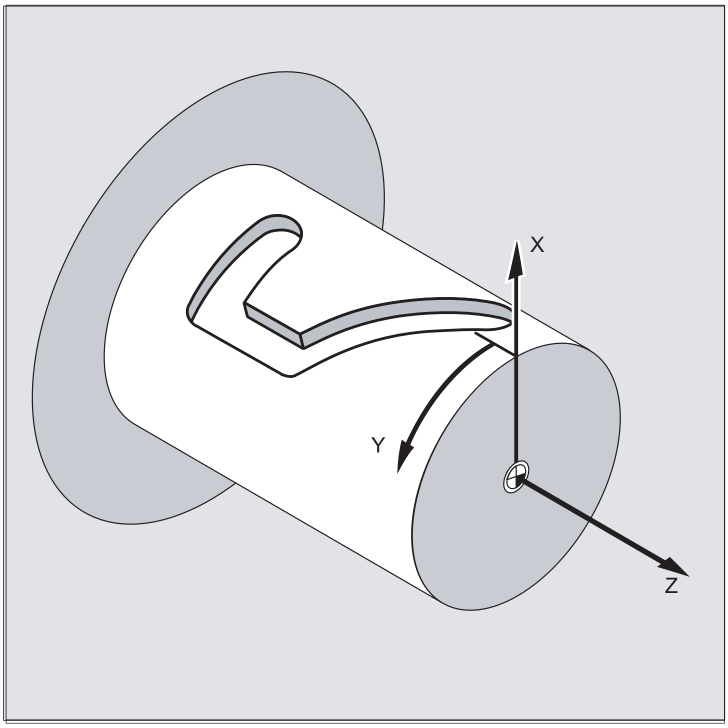


Figure 6-7 TRACYL cylinder surface transformation

Note

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

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

One or two linear axes (axis configuration 1)

One linear axis

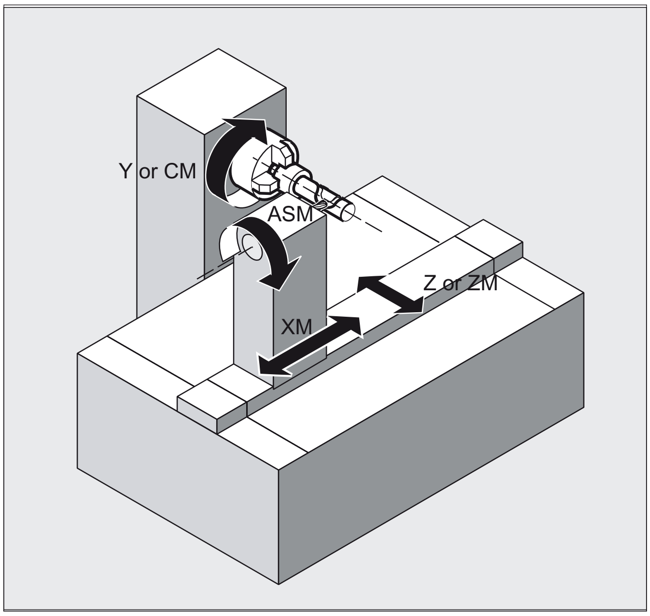
For a machine kinematic with only one linear axis (X), only grooves parallel to the periphery of the cylinder can be generated.

Two linear axes

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

The angularity of the groove edges for groove widths larger than the tool diameter:

- Grooves parallel to the circumference ⇒ groove edges parallel to each other
- Grooves parallel to the rotary axis ⇒ groove edges at angle to each other



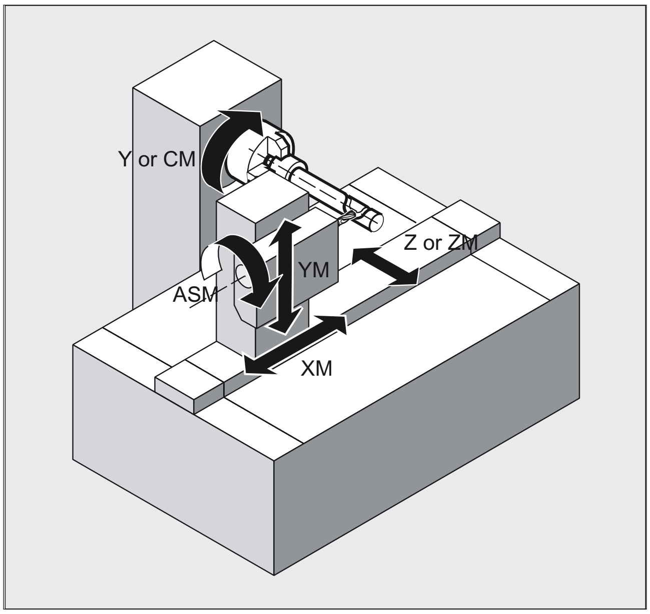
XM	Infeed axis perpendicular to the turning center
ZM	Linear axis parallel to the turning center
Y / CM	Transformatory Y axis / rotary axis
ASM	Working spindle

Figure 6-8 Without groove side correction

Three linear axes (axis configuration 2)

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

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



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

Figure 6-9 With groove side correction

6.3.1 Preconditions

Maximum number of TRACYL transformations per channel

The following machine data can be used to define transformation data records in a channel:

- $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>
- $MC_TRAFO_TYPE_<n>
- $MC_TRAFO_AXES_IN_<n>

with n = 1, 2, 3, ... max. number of transformation data records

Maximum two data records can be used for the `TRACYL`, $MC_TRAFO_TYPE_<n> = 512, 513 or 514 transformation in a channel.

Machine data for TRACYL transformation

A TRACYL transformation is parameterized using the following machine data:

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

with n = 1, 2, ... max. number of possible `TRACYL` transformations

Sample axis configuration

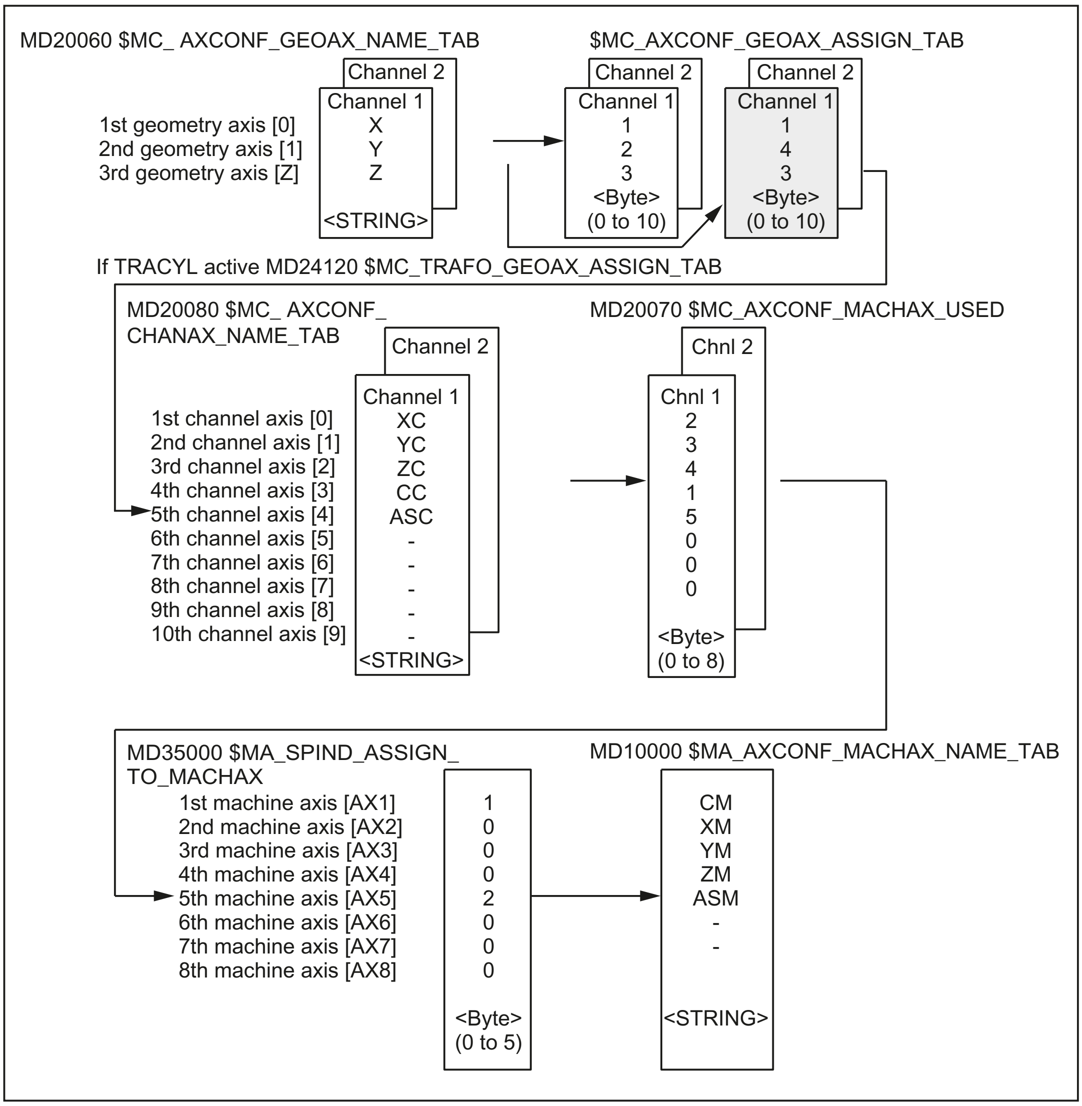


Figure 6-10 Axis configuration for the example in Figure "Machining grooves on a cylinder surface with X-Y-Z-C kinematics"

The configurations highlighted in the figure above apply when TRACYL is active.

Geometry axis names

- MD20050 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"
- MD20050 $MC_AXCONF_GEOAX_NAME_TAB[1]="Y"
- MD20050 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"

Assignment of geometry axes to channel axes

TRACYL not active

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

TRACYL active

- $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>[0] = 1
- $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>[1] = 4
- $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>[2] = 3

Channel axis names

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0]="XC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1]="YC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2]="ZC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="CC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4]="ASC"

Assignment of channel axes to machine axes

MD20070 $MC_AXCONF_MACHAX_USED[0] = 2

MD20070 $MC_AXCONF_MACHAX_USED[1] = 3

MD20070 $MC_AXCONF_MACHAX_USED[2] = 4

MD20070 $MC_AXCONF_MACHAX_USED[3] = 1

MD20070 $MC_AXCONF_MACHAX_USED[4] = 5

Identification of spindles

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0] = 1 (spindle)

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1] = 0 (axis)

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2] = 0 (axis)

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3] = 0 (axis)

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[4] = 2 (spindle)

Machine axis name

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]="CM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]="XM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]="YM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="ZM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4]="ASM"

See also

TRACYL cylinder surface transformation (option) (Page 366)

6.3.2 Specific settings

Setting of the transformation type

The transformation type is set transformation-data-record-specific via:

- $MC_TRAFO_TYPE_<n> = <transformation type>

The user must specify the transformation type for the transformation data blocks. For `TRACYL` a VALUE of 512 must be set for axis configuration 1 and a value of 513 for axis configuration 2 or 514 for transformation with additional linear axis.

The transformation with additional linear axis can be activated with and without groove side offset (see Section "Switch on (Page 375)").

Transformation type 514 without groove side offset

Cylinder surface curve transformation TRAFO_TYPE_n = 514

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

The existing settings for MD10000 $MC_TRAFO_GEOAX_ASSIGN_TAB_n apply.

Grooves with groove side offset

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

Axis image

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

Transformation geometry axes

Three (or four) channel axis numbers must be specified for `TRACYL` :

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

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

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_n

Grooves without groove wall offset

For transformation type 514 the following indices apply to MD24110 $MC_TRAFO_AXES_IN_n[ ].

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

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

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

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

Rotational position

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

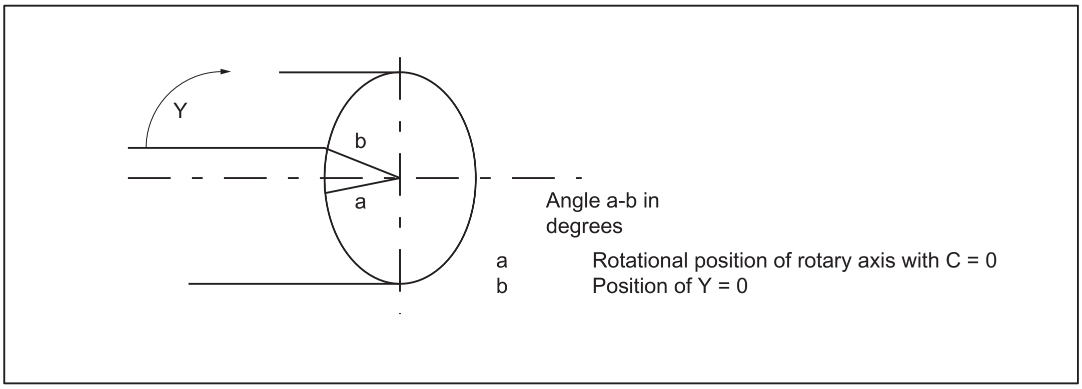


Figure 6-11 Center of axis rotation in the peripheral cylinder surface

MD24800 TRACYL_ROT_AX_OFFSET_<t>

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

MD24800 $MC_TRACYL_ROT_AX_OFFSET_t=...°

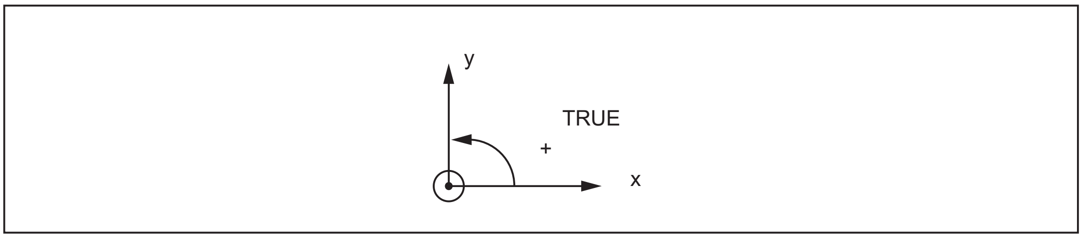
In this case, "t" is substituted by the number of TRACYL transformations declared in the transformation data blocks (t may not be greater than 2).

Direction of rotation

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

TRACYL_ROT_SIGN_IS_PLUS_<t>

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



MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_t=TRUE

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

Replaceable geometry axes

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

- MD22534 $MC_TRAFO_CHANGE_M_CODE

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

  Note

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

References:

Function Manual Basic Functions; Coordinate Systems, Axis Types, Axis Configurations, Workpiece-related Actual-Value System, External Zero Offset (K2)

Position of tool zero

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

MD24820 TRACYL_BASE_TOOL_<t>

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

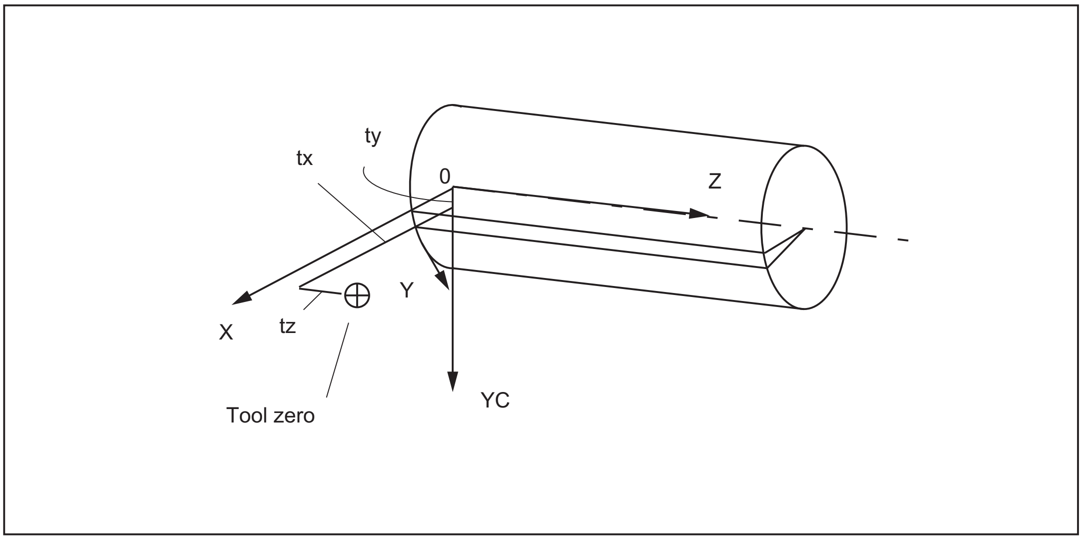


Figure 6-12 Position of tool zero in relation to machine zero

Example

MD24820 $MC_TRACYL_BASE_TOOL_<t>[0] = tx

MD24820 $MC_TRACYL_BASE_TOOL_<t>[1] = ty

MD24820 $MC_TRACYL_BASE_TOOL_<t>[2] = tz

Where <t> = number of TRACYL transformations defined in the transformation data records

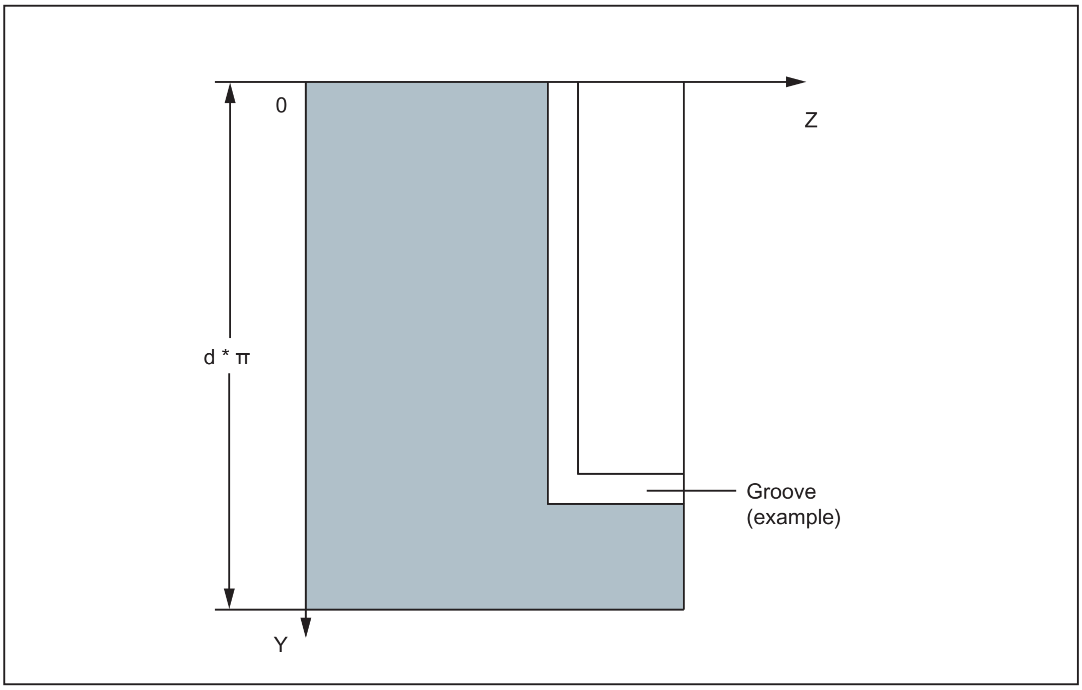


Figure 6-13 Cylinder coordinate system

See also

TRACYL cylinder surface transformation (option) (Page 366)

6.3.3 Switch on

The TRACYL transformation is activated in the part program or synchronous action using the `TRACYL` command.

Syntax

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

Meaning

`TRACYL(<d>)`:	Activate TRACYL with TRACYL data set 1 and working diameter d	
`TRACYL(<d>, <n>)`:	Activate TRACYL with TRACYL data set n and working diameter d	
`<d>`:	Reference or working diameter.	
	Range of values:	>1
`<n>`:	TRACYL data set number (optional)	
	Range of values:	1, 2
`<k>`:	The parameter is only relevant for transformation type 514 (optional). • k = 0: without groove side correction • k = 1: with groove side correction If the parameter is not specified, then the parameterized basic position applies: $MC_TRACYL_DEFAULT_MODE_<n>	

6.3.4 Deactivation

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

- Deactivate transformation: `TRAFOOF`
- Activate another transformation, e.g. `TRAANG`, `TRANSMIT`, `TRAORI`

6.3.5 Boundary conditions

Selection/deselection

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

Tool change

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

Frame

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

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

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

$MC_TRACYL_ROT_AX_FRAME_<t> = 1

Note

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

References:

Function Manual Basic Functions; "Co-ordinate Systems, Frames" (K2)

Function restriction for transformation axes

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

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

Manual traversal in JOG

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

Interrupt part program

- Mode change from AUTOMATIC to JOG

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

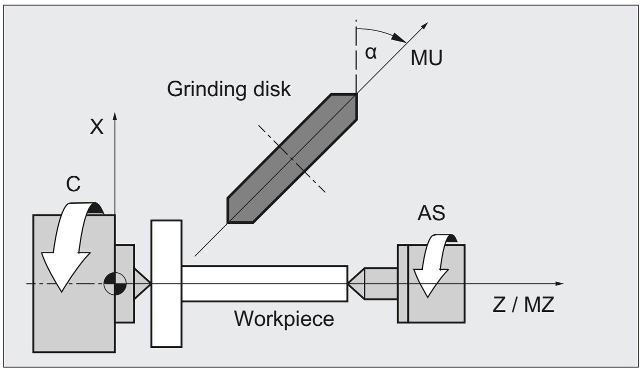
  > **⚠ WARNING**
  >
  > **Risk of collision**
  >
  > No monitoring for collisions takes place.
  >
  > The operator is responsible for ensuring that the tool can be re-positioned without any difficulties.

- START after RESET

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

6.4 TRAANG oblique angle transformation (option)

The angle transformation or "inclined axis" transformation permits the programming in the perpendicular workpiece coordinate system (WCS) on machines with inclined machine axes. A typical axis orientation for grinding machines:



X	Geometry axis
Z	Geometry axis
MZ	Machine axis
MU	Machine axis
α	Angle of inclined axis

Note

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

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

6.4.1 Preconditions

Maximum number of TRAANG transformations per channel

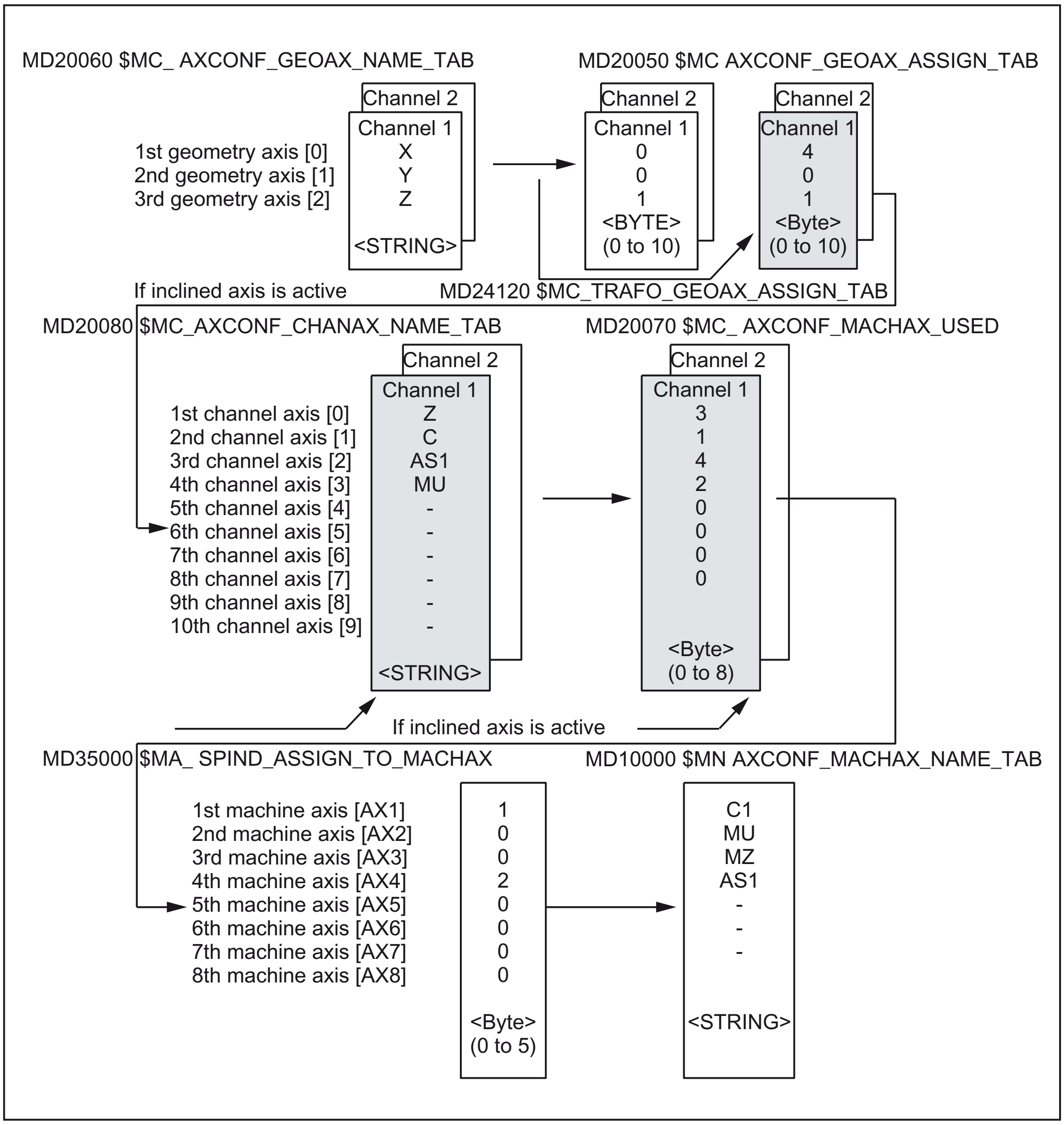
The following machine data is used to define the channel-specific transformation data records:

- $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>
- $MC_TRAFO_TYPE_<n>
- $MC_TRAFO_AXES_IN_<n>

with n = 1, 2, 3, ... max. number of transformation data records

Maximum two transformation data records of these may be used for transformation `TRAANG`, $MC_TRAFO_TYPE_<n> = 1024.

Example of an axis configuration



The configurations highlighted in the figure above apply when `TRAANG` is active.

Transformation channel axes

- MD24110 $MC_TRAFO_AXES_IN_1[0]=4; channel axis number of the inclined axis
- MD24110 $MC_TRAFO_AXES_IN_1[1]=1; channel axis number of the longitudinal axis
- MD24110 $MC_TRAFO_AXES_IN_1[2]=0 ; ---

Assignment of geometry axes to channel axes

TRAANG active

- MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>[0] = 1
- MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>[1] = 6
- MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>[2] = 3

See also

TRAANG oblique angle transformation (option) (Page 378)

6.4.2 Specific settings

Angle between longitudinal axis and inclined axis

- MD24700 $MC_TRAANG_ANGLE_<m> = <angle>

with -90° < angle < 90° , without 0°

The angle is counted positively in the clockwise direction starting at X (see Section "TRAANG oblique angle transformation (option) (Page 378)": Angle α).

Base offset of the tool zero

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

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

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

Note

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

Optimization of velocity control

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

Speed margin

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

MD24720 $MC_TRAANG_PARALLEL_VELO_RES_<n> = <value>

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

Acceleration margin

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

$MC_TRAANG_PARALLEL_ACCEL_RES_<n> = <value>

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

M code for transformation change

The specified M code is output for switching the transformation:

- MD22534 $MC_TRAFO_CHANGE_M_CODE = <value>

> ⚠ WARNING
>
> **No M code is output.**
>
> The values 0 to 6, 17 and 30 are not output to the PLC.

6.4.3 Switch on

The TRAANG transformation is activated in the part program or synchronous action using the `TRAANG` command.

Syntax

```
TRAANG
TRAANG([<α>][, <n>])
```

Meaning

`TRAANG`:	Activate TRAANG with data set 1 and last valid angle α	
`TRAANG(, <n>)`:	Activate TRAANG with data set n and last valid angle α	
`TRAANG(<α>)`:	Activate TRAANG with data set 1 and angle α	
`TRAANG(<α>, <n>)`:	Activate TRAANG with data set n and angle α	
`<α>`:	Angle of the inclined axis (optional) **Note** Without specifying the angle, the value parameterized in the machine data: $MC_TRAANG_ANGLE_<n>, with n = data set number is valid	
	Range of values:	-90° < α < + 90°
`<n>`:	Data set number (optional)	
	Range of values:	1, 2

6.4.4 Deactivation

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

- Deactivate transformation: `TRAFOOF`
- Activate another transformation, e.g. `TRACYL`, `TRANSMIT`, `TRAORI`

6.4.5 Boundary conditions

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

Selection and deselection

- An intermediate motion block is not inserted (phases/radii).
- A spline block sequence must be terminated.
- Tool radius compensation must be deselected.
- The current frame is deselected by the control system.

  (corresponds to programmed G500).
- An active working area limitation is deselected by the control for the axes affected by the transformation (corresponds to programmed `WALIMOF`).
- An activated tool length compensation is included in the transformation by the control.
- Continuous path control and rounding are interrupted.
- DRF offsets must have been deleted by the operator.
- All axes specified in machine data MD24110 $MC_TRAFO_AXES_IN_n must be synchronized on a block-related basis (e.g. no traversing instruction with POSA...).

Tool change

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

Frame

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

Extensions

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

For additional information on these frame corrections with transformations, see:

References:

Function Manual, Basic Functions; Axes, Coordinate Systems, Frames (K2)

Exceptions

Axes affected by the transformation cannot be used

- As a preset axis (alarm)
- To approach the fixed point (alarm)
- For referencing (alarm)

Velocity control

The velocity monitoring function for TRAANG is implemented as standard during preprocessing.

Monitoring and limitation in the main run are activated:

- In AUTOMATIC mode, if a positioning or oscillation axis has been programmed that is involved in the transformation.
- On changeover to JOG mode

The monitoring function is transferred again from the main run to block preprocessing if the preprocessing is re-synchronized with the main run (currently, for example, on changeover from JOG to AUTOMATIC).

The velocity monitoring function in preprocessing utilizes the dynamic limitations of the machine better than the monitoring function in the main run.

This also applies to machines on which oblique machining operations are performed.

6.4.6 Programming (G05, G07)

Function

The following functions are available:

- Position programming and display in the Cartesian coordinate system
- Cartesian calculation of tool offset and work offset
- Programming of angles for the inclined axis in the NC program
- Approach the starting position for inclined plunge cutting (G07)
- Inclined plunge cutting (G05)

Programming

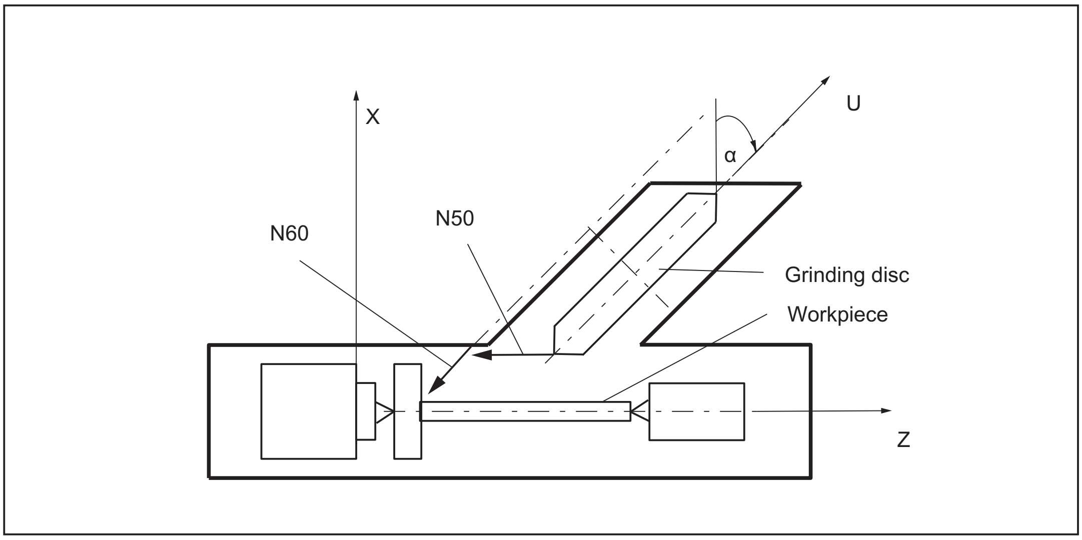


Figure 6-14 Machine with inclined infeed axis

Example:

Program code	Comment
N...	; Program axis for inclined axis
N50 G07 X70 Z40 F4000	; Approach starting position
N60 G05 X70 F100	; Oblique plunge-cutting
N...	

Supplementary conditions

- It is only meaningful to select the function "Cartesian PTP travel" in JOG mode (motion according to G05) if a transformation is active (`TRAANG`). Note the value set in MD20140 $MC_TRAFO_RESET_VALUE.
- `REPOS` offsets must be traversed back in JOG mode in the Cartesian coordinates while "PTP travel" is not active.
- The working area limitation is monitored in JOG mode with active "PTP travel". The axis is braked before overrunning the working area limitation. If "PTP travel" is not active, the axis can be traversed right up to the operating range limit.

See also

Cartesian PTP travel (Page 398)

6.5 Chained transformations

Introduction

It is possible to chain the kinematic transformation described here, with an additional transformation of the type "Inclined axis":

- `TRANSMIT`
- `TRACYL`
- `TRAANG` (oblique axis)

as described in

References:

Function Manual, Special Functions; 3-Axis to 5-Axis Transformation (F2)

.

- Orientation transformations
- Universal milling head

Applications

The following is a selection from the range of possible chained transformations:

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

Note

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

MD10000: AXCONF_MACHAX_NAME_TAB

MD20080: AXCONF_CHANAX_NAME_TAB

MD20060: AXCONF_GEOAX_NAME_TAB

Besides this, no unequivocal assignments exist.

Axis configuration

The following configuration measures are necessary for a chained transformation:

- Assignment of names to geometry axes
- Assignment of names to channel axes

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

Number of transformations

Up to ten transformation data blocks can be defined for each channel in the system. The machine data names of these transformations begin with $MC_TRAFO .. and end with ... _n, where n stands for a number between 1 and 10.

Number of chained transformations

Within the maximum of 10 transformations of a channel, a maximum of **two chained** transformations may be defined.

Transformation sequence

When configuring the machine data, the data concerning the single transformations (that may also become part of chained transformations) must be specified before the data concerning the chained transformations.

Chaining sequence

With chained transformations the second transformation must be "inclined axis" (`TRAANG`).

Chaining direction

The BCS is the input for the first of the transformations to be chained; the MCS is the output for the second one.

Supplementary conditions

The supplementary conditions and special cases indicated in the individual transformation descriptions are also applicable for use in chained transformations.

6.5.1 Activating chained transformations

TRACON

A chained transformation is activated by:

TRACON(trf, par)

- trf:

  Number of the chained transformation: 0 or 1 for first/only chained transformation. If nothing is programmed here, then this has the same meaning as specifying value 0 or 1, i.e., the first/only transformation is activated – 2 for the second chained transformation. (Values not equal to 0 - 2 generate an error alarm).
- par

  One or more parameters separated by a comma for the transformations in the chain expecting parameters, for example, the angle of the inclined axis. If parameters are not set, the defaults or the parameters last used take effect. Commas must be used to ensure that the specified parameters are evaluated in the sequence in which they are expected, if default settings are to be effective for previous parameters. In particular, a comma is required before at least one parameter, even though it is not necessary to specify trf. For example: TRACON(, 3.7).`TRACON`(, 3.7).

If another transformation was previously activated, it is implicitly disabled by means of `TRACON`().

6.5.2 Switching off a chained transformation

TRAFOOF

A chained transformation is switched off with `TRAFOOF` just like any other transformation.

6.5.3 Special characteristics of chained transformations

Tool data

A tool is always assigned to the first transformation in a chain. The subsequent transformation then behaves as if the active tool length were zero. Only the basic tool lengths set in the machine data (_BASE_TOOL_) are valid for the **first** transformation in the chain.

Example

The chapter "Chained transformations" contains configuration examples for single transformations and the transformation chains created from them.

See also

Chained transformations (Page 387)

6.5.4 Persistent transformation

Function

A persistent transformation is always active and has a relative effect to the other explicitly selected transformations. Other selected transformation are computed as the first chained transformation in relation to the persistent transformation.

Transformations such as TRANSMIT that must be selected in relation to the persistent transformation must be parameterized in a chain with the persistent transformation by means of TRACON. It is the first chained transformation rather than the TRACON transformation which is programmed in the part program.

For additional programming tips see

References:

Programming Manual, Job Planning; Transformations "Chained Transformation"

Selection and deselection

Persistent transformation is selected via the following machine data:

MD20144 $MC_TRAFO_MODE_MASK, Bit 0 = 1

MD20144 $MC_TRAFO_RESET_VALUE defines persistent transformation.

MD20140 $MC_TRAFO_RESET_VALUE=Number of the transformation data set of the persistent transformation

In addition the following must be set (i.e. noted):

MD20110 $MC_RESET_MODE_MASK

Bit 0 = 1 (Bit 7 is evaluated)

Bit 7 =0 (MD20140 $MC_TRAFO_RESET_VALUE determines the transformation data set)

MD20112 $MC_START_MODE_MASK (MD20140 $MC_TRAFO_RESET_VALUE)

MD20118 $MC_GEOAX_CHANGE_RESET= TRUE (i.e. geometry axes are reset).

If this additional data is not parameterized correctly,

alarm 14404 is generated.

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

Effects on HMI operation

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

`TRACON` on HMI

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

`TRACOOF` on HMI

In accordance with the `TRAFOOF` programming instruction **no** transformation is displayed in the G code list on the HMI user interface. System variables $P_TRAFO and $AC_TRAFO therefore return a value of 0, the persistent transformation is operative and the BCS and MCS coordinate systems do not coincide. The displayed MCS position always refers to the actual machine axes.

System variable

New system variables return the transformation types of the active chained transformations.

Description	NCK variable
no transformation active: 0 one transformation active: Type of 1st chained transformation with TRACON, or type of active transformation if not TRACON	$P_TRAFO_CHAIN[0]
no transformation active: 0 one transformation active: Type of 2nd chained transformation with TRACON	$P_TRAFO_CHAIN[1] $AC_TRAFO_CHAIN[1]
are only used if more than 2 transformations are chained. These variables presently only return 0.	$P_TRAFO_CHAIN[2] $AC_TRAFO_CHAIN[2] and $P_TRAFO_CHAIN[3] $AC_TRAFO_CHAIN[3]

Display persistent transformation:

$P_TRAFO_CHAIN[0], $AC_TRAFO_CHAI[0]

These settings allow an active transformation to be displayed reliably in the part program or in cycles.

Difference between a TRACON and the other transformations:

$P_TRAFO, $AC_TRAFO if no transformation is active, or $P_TRAFO_CHAIN[1], $AC_TRAFO_CHAI[1] is interrogated for a value other than zero.

Frames

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

JOG

The persistent transformation remains in effect when traversing with JOG.

Supplementary conditions

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

A `RESET` command still deselects any active transformation completely; the persistent transformation is selected again. The persistent transformation is not reselected under error conditions. A corresponding alarm is generated to indicate the error constellation.

Alarm 14401 or 14404 can be activated when `TRAANG` is the persistent transformation. When the persistent transformation is active, other transformation alarms may generated in response to errors depending on the transformation type selected.

The transformation is deselected implicitly during referencing. A `RESET` or `START` command must be issued after referencing, in order to reselect the persistent transformation.

Example

For a lathe with an inclined additional Y axis, the transformation of the inclined axis should be part of the machine configuration and therefore does not have to be considered by the programmer. With `TRACYL` or `TRANSMIT` transformations are selected, which must then include the `TRAANG`. When the programmed transformations are deactivated, `TRAANG` is automatically activated again. In the `HMI` operator interface `TRACYL` or `TRANSMIT` is displayed.

Machine data for a turning machine with Y1 axis inclined in relation to X1 but perpendicular to Z1.

CANDATA (1)

; Kinematic without transformations

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "Y2"

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

; Data for TRAANG

MD24100 $MC_TRAFO_TYP_1 = 1024; TRAANG Y1 axis inclined to X1, perpendicular to Z1

MD24110 $MC_TRAFO_AXES_IN_1[0]=2

MD24110 $MC_TRAFO_AXES_IN_1[1]=1

```
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3
MD24110 $MC_TRAFO_AXES_IN_1[3] = 0
MD24110 $MC_TRAFO_AXES_IN_1[4] = 0
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3
MD24700 $MC_TRAANG_ANGLE_1 = 60
MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
 ; Definition of persistent transformation
MD20144 $MC_TRAFO_MODE_MASK = 1
MD20140 $MC_TRAFO_RESET_VALVUE= 1
MD20110 $MC_RESET_MODE_MASK = 'H01'
MD20112 $MC_START_MODE_MASK = 'H80'
MD20140 $MC_TRAFO_RESET_VALUE
MD20118 $MC_GEOAX_CHANGE_RESET= TRUE
 ; Data for TRANSMIT, TRACYL
MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 ; also 2, causes alarm 21617
MD24200 $MC_TRAFO_TYP_2 = 257
MD24210 $MC_TRAFO_AXES_IN_2[0] = 1
MD24210 $MC_TRAFO_AXES_IN_2[1] = 4
MD24210 $MC_TRAFO_AXES_IN_2[2] = 3
MD24210 $MC_TRAFO_AXES_IN_2[3] = 0
MD24210 $MC_TRAFO_AXES_IN_2[4] = 0
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] =1
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] =4
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] =3
MD24300 $MC_TRAFO_TYP_3 = 514
MD24310 $MC_TRAFO_AXES_IN_3[0] = 1
MD24310 $MC_TRAFO_AXES_IN_3[1] = 4
MD24310 $MC_TRAFO_AXES_IN_3[2] = 3
MD24310 $MC_TRAFO_AXES_IN_3[3] = 0
MD24310 $MC_TRAFO_AXES_IN_3[4] = 0
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =1
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =4
```

```
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =3
 ; Data for TRACON
 ; TRACON chaining TRANSMIT 514/TRAANG(Y1 axis inclined in relation to X1)
MD24400 $MC_TRAFO_TYP_4 = 8192
MD24995 $MC_TRACON_CHAIN_1[0] = 3
MD24995 $MC_TRACON_CHAIN_1[1] = 1
MD24995 $MC_TRACON_CHAIN_1[2] = 0
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =4
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3
 ; TRACON chaining TRANSMIT 257/TRAANG(Y1 axis inclined in relation to X1)
MD24430 $MC_TRAFO_TYP_5 = 8192
MD24996 $MC-TRACON_CHAIN_2[0] = 2
MD24996 $MC-TRACON_CHAIN_2[1] = 1
MD24996 $MC_TRACON_CHAIN_2[2] = 0
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =4
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3
M17
 ; matching part program:
$TC_DP1[1,1]=120 ; tool type
$TC_DP2[1,1] = 0
$TC_DP3[1,1]=3 ; length compensation vector
$TC_DP4[1,1]=25
$TC_DP5[1.1] =5
$TC_DP6[1,1]= 2; Radius; tool radius
 ; transformation change:
N1000 G0 X0 Y=0 Z0 A80 G603 SOFT G64
N1010 N1020 X10 Y20 Z30 ; TRAANG(,1) not possible, since automatically selected
N1110 TRANSMIT(1) N1120 X10 Y20 Z30N1130 Y2=0 ; TRACON(2) not necessary, since translated automatically
N1210 TRAFOOF ; TRAANG(,1) not necessary, since translated automatically
N1220 X10 Y20 Z30
M30
```

6.5.5 Axis positions in the transformation chain

Function

System variables having the following content are provided for machines with system or OEM transformations, especially for chained transformations (TRACON):

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

The following must be observed with reference to the control system responses:

- POWER ON

  The encoder position has the value 0 for not-referenced axes. The encoder actual values are inverse-transformed accordingly for the $VA variables.

- RESET

  An active transformation can change in RESET, which has a direct influence on the values of the system variables. An active transformation which is active again after RESET, is deactivated for a short duration and then reactivated. This has a direct influence on the position variables. The values of variables can change.

  Via the variable:

  $AC_STAT == 0

  this status can be queried in synchronous actions.

$AA_ITR[<axis>, <transformer layer>]

The $AA_ITR[ax,n] variable determines the setpoint position of an axis at the output of the nth chained transformation.

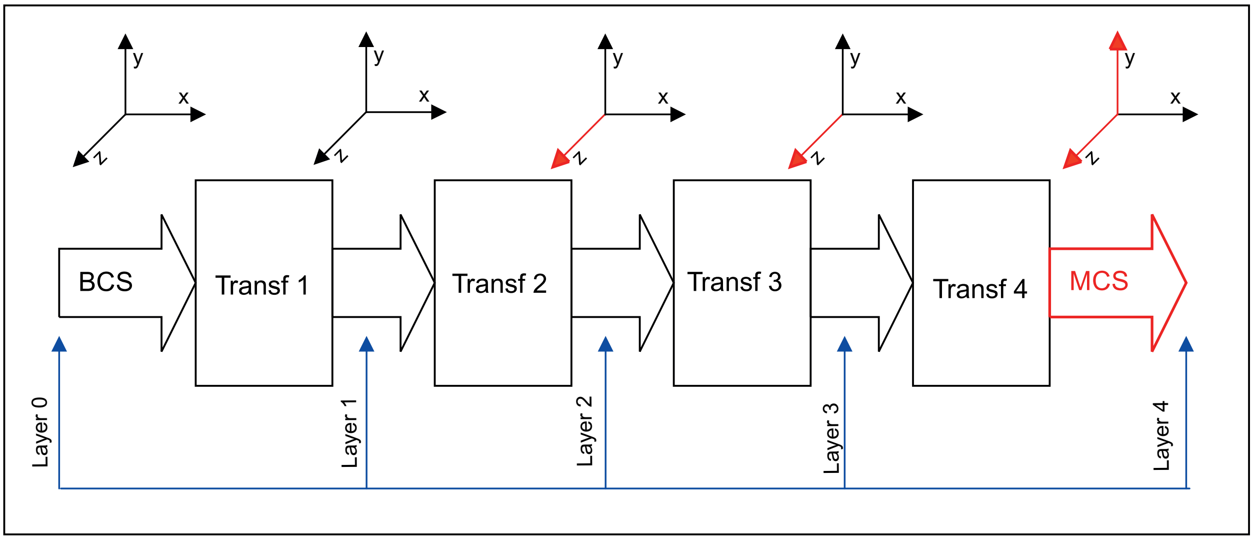


Figure 6-15 Transformer layer

Transformer layer

The 2nd index of the variable corresponds to the transformer layer in which the positions are tapped:

- Transformer layer 0: The positions correspond to the BCS positions, i.e.:

  $AA_ITR[x,0] == $AA_IB[x]
- Transformer layer 1: Setpoint positions at output of 1st transformation
- Transformer layer 2: Setpoint positions at output of 2nd transformation
- Transformer layer 3: Setpoint positions at output of 3rd transformation
- Transformer layer 4: Setpoint positions at output of 4th transformation, i.e.

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

If one or more transformations of the transformer chain are missing, the highest layers continue to deliver the same values. If, e.g. Transformer 3 and Transformer 4 are missing, this corresponds to:
$AA_ITR[x,2] = $AA_ITR[x,3] = $AA_ITR[x,4] = $AA_IM[x]

If the transformations are shut off via `TRAFOOF` or in RESET, the Layers 0 to 4 fuse together and the variable always delivers the BCS value (Layer 0) in this case.

Axis

Either a geometry, channel or a machine axis name is permissible as the 1st index of the variable. The assignment of the channel axes to the geometry axes corresponding to the 0 layer takes place during the programming of geometry axis name in each transformer layer. Using geometry axis names is meaningful only if the geometry axes are not switched over. Otherwise it is always better to use channel axis names.

$AA_IBC[<axis>]

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

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

$VA_ITR[<axis>, <transformer layer>]

The variable $VA_ITR[ax,n] determines the encoder position of an axis at the output of the nth chained transformation.

$VA_IBC[<axis>]

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

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

$VA_IW[<axis>]

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

$VA_IB[<axis>]

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

Note

VA_ITR, VA_IBC, $VA_IW, $VA_IB

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

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

6.6 Cartesian PTP travel

Function

This function can be used to approach a Cartesian position with a synchronized axis movement.

It is particularly useful in cases where, for example, the position of the joint is changed, causing the axis to move through a singularity.

When an axis passes through a singularity, the feed velocity would normally be reduced or the axis itself overloaded.

Note

MD24100 $MC_TRAFO_TYPE_1 must be set to the transformation type described in TE4.

The function can only be used meaningfully in conjunction with an active transformation. Furthermore, the "Cartesian PTP travel" function may only be used in conjunction with the G0 and G1 commands. Alarm 14144 "PTP travel not possible" is otherwise output.

When PTP is active, axes in the transformation, e.g. which are traversed using POS, cannot be simultaneously positioning axes. Alarm 17610 is activated to prevent this error.

Activation

The function is activated when the PTP command is programmed.

The function can be deactivated again with the CP command. Both these commands are contained in G group 49.

- PTP command: The programmed Cartesian position is approached with a synchronized axis motion (PTP=point-to-point)
- CP command: The programmed Cartesian point is approached with a path movement (default setting), (CP=continuous path)
- PTPG0 command: The programmed Cartesian PTP motion is performed automatically with each G0 block. The CP command is then set again.

Power On

After `Power on` traversing mode CP is automatically set for axis traversal with transformation. MD20152 $MC_GCODE_RESET_VALUES[48] can be used to switch the default setting to cartesian PTP travel.

Reset

MD20152 $MC_GCODE_RESET_MODE[48] (group 49) defines which setting is active after RESET/end of part program.

- MD=0: Settings are effected in accordance with machine data

  MD20150 $MC_GCODE_RESET_VALUES[48]
- MD=1: Active setting remains valid

Selection

The setting MD20152 $MC_GCODE_RESET_MODE[48] =0, with MD20150 $MC_GCODE_RESET_VALUES[48] can activate the following:

- MD=2:

  Cartesian PTP travel as previously or
- MD=3:

  PTPG0, traverse only G0 blocks with PTP automatically and then switch over to CP again

Supplementary conditions

The following should be noted with respect to tool movement and collision:

- As the PTP command can produce significantly different tool movements to the CP command, any pre-existing subprograms which have been written independently of the active transformation must be adapted to take account of the risks of collision when TRANSMIT is active. This applies particularly in the case of command PTPG0.
- Machine axes always traverse the shortest possible path in response to TRANSMIT and PTP. Minor displacements in the block end point can cause the rotary axis to rotate by -179.99° instead of + 179.99°, even though the block end point has hardly changed.

The following combinations with other NC functions are not legal:

- No tool radius compensation (TRC) may be active with PTP.

  G0 and G41 do not exclude each other in principle. However, an active PTP generates different contours to those computed for the TRC, resulting in the activation of a TRC alarm.
- With PTPG0 , for active tool radius compensation (TRC), traverse is by CP.

  Since G0 and G41 do not exclude each other, switch-over to CP is done automatically when tool radius compensation is active. The radius compensation therefore works on the basis of clearly defined contours.
- PTP does not permit smooth approach and retraction (SAR).

  SAR requires a contour in order to construct approach and retraction motion. This information is not available with PTP.

- With PTPG0, CP travel is used for smooth approach and retraction (SAR).

  SAR requires a contour in order to construct approach and retraction motion and to be able to lower and raise tangentially. The blocks required for this purpose are therefore traversed with the CP command. The G0 blocks up to the actual approach contour are executed with PTP and therefore quickly. The same applies to the retract blocks.
- PTP does not permit cutting cycles like CONTPRON, CONTDCON

  Stock removal cycles require a contour to construct the cut segmentation. This information is not available with PTP. Alarm 10931 "Error in cut compensation" is generated in response.
- When PTPG0 is selected, the CP command is applied in cutting cycles such asCONTPRON, CONTDCON. Stock removal cycles require a contour to construct the cut segmentation. The blocks required for this purpose are traversed with the CP command.
- Chamfer and rounding are ignored.
- An axis override in the interpolation must not change during the PTP contour section. This applies, for example, to LIFTFAST, fine tool offset, coupled motion TRAILON and tangential follow-up TANGON.

In PTP blocks

- compressor is automatically deselected because it is not compatible with PTP.
- G643 is automatically switched over to G642.
- Transformation axes must not be configured simultaneously as positioning axes.

Special features

Please take account of the following basic rules with respect to the basic coordinate system:

- Smoothing G642 is always interpreted in the machine coordinate system and not (as usual) in the cartesian basic coordinate system.
- G641 determines the smoothing action as a function of the fictitious path calculated from the machine axis coordinates.
- An F value input with G1 refers to the fictitious path calculated from the machine axis coordinates.

Block search

TRANSMIT during block search can result in different machine axis positions for the same Cartesian position, if a program section is executed with block search.

Alarms

An illegal action, which may result in a conflict, is rejected with the following alarms:

Alarm 14144: If TRC is selected or activated in PTP. Likewise in PTP with soft approach and retraction (SAR) or PTP without the required G0 and G1 blocks.

Alarm 10753: With PTPG0 and active TRC an internal switch-over to CP is done in order to allow the tool radius correction to be performed correctly.

Alarm 10754: Still possible in case of conflict.

Alarm 10778: Still possible in case of conflict.

Alarm 10744: With PTPG0, CP travel is used for smooth approach and retraction (SAR), in order to ensure correct processing of soft approach and retraction.

Alarm 10746: Still possible in case of conflict.

Alarm 17610: Transformation axes must not be configured simultaneously as positioning axes traversed by means of POS.

Note

For further information about programming plus programming examples, please see:

References:

Programming Manual, Job Planning; Section Transformations, "Cartesian PTP Travel"

6.6.1 Programming of position

Generally speaking, a machine position is not uniquely defined solely by a position input with Cartesian coordinates and the orientation of the tool. Depending on the kinematics of the relevant machine, the joint may assume up to 8 different positions. These joint positions are specific to individual transformations.

STAT address

A Cartesian position must be convertible into unique axis angles. For this reason, the position of the joints must be entered in the `STAT` address.

The `STAT` address contains a bit for every possible setting as a binary value. The meaning of these bits is determined by the relevant transformation.

As regards the transformations contained in the publication "Handling Transformation Package (TE4)", the bits are assigned to different joint positions, as shown in the figure above.

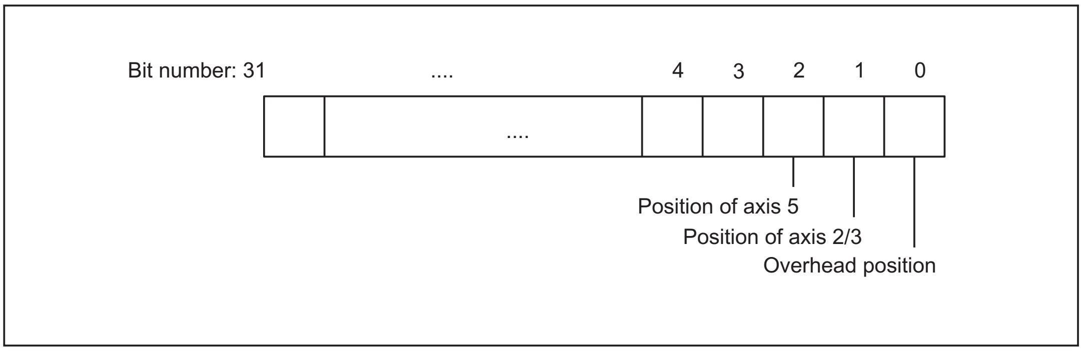


Figure 6-16 Position bits for Handling Transformation Package

---

Note

It is only meaningful to program the `STAT` address for "Cartesian PTP travel", since changes in position are not normally possible while an axis is traversing with active transformation. The starting point position is applied as the destination point for traversal with the CP command.

6.6.2 Overlap areas of axis angles

TU address

In order to approach axis angles in excess of ±180° without ambiguity, the information must be programmed in the TU (turn) address. The TU address thus represents the sign of the axis angles. This allows an axis angle of $|\theta| < 360°$ to be traversed without ambiguity.

Variable TU contains a bit, which indicates the traversing direction for every axis involved in the transformation.

- TU bit=0: $0° \leq \theta < 360°$
- TU bit=1: $360° < \theta < 0°$

The TU bit is set to 0 for linear axes.

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

If no TU is programmed for a position, the axis always traverses via the shortest possible route.

6.6.3 Examples of ambiguities of position

The kinematics for a 6axis joint have been used to illustrate the ambiguities caused by different joint positions.

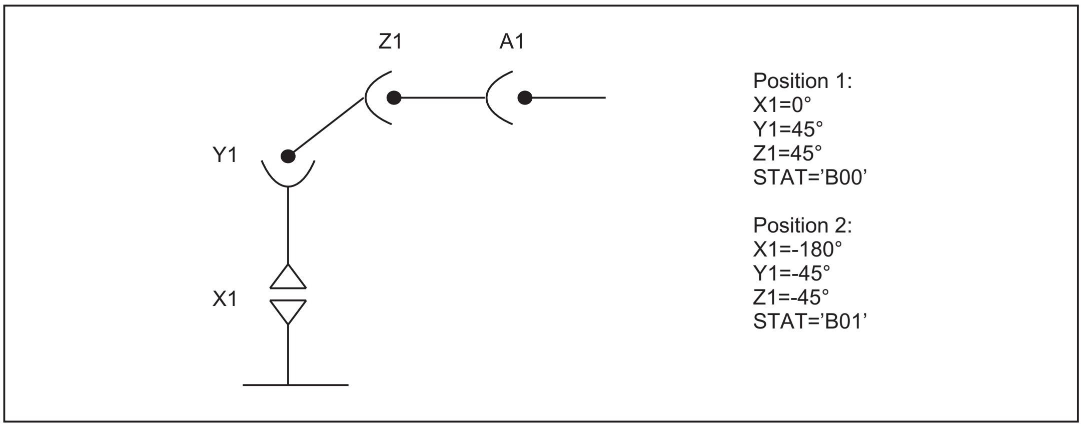


Figure 6-17 Ambiguity in overhead area

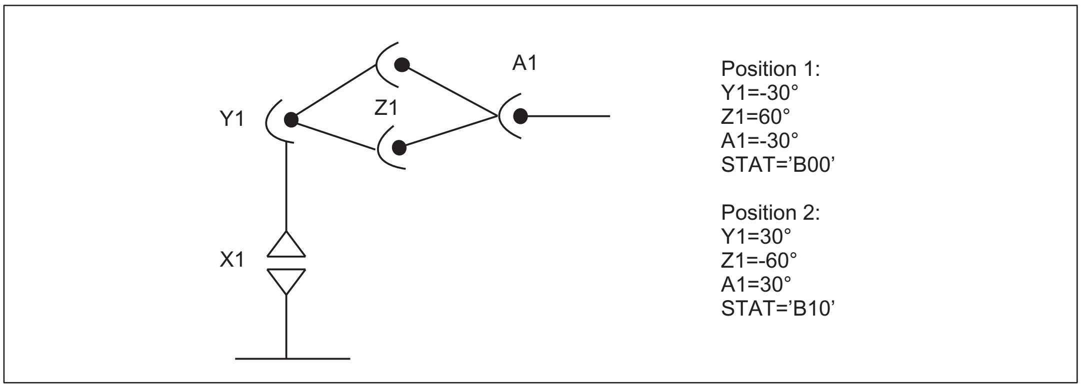


Figure 6-18 Ambiguity of top or bottom elbow

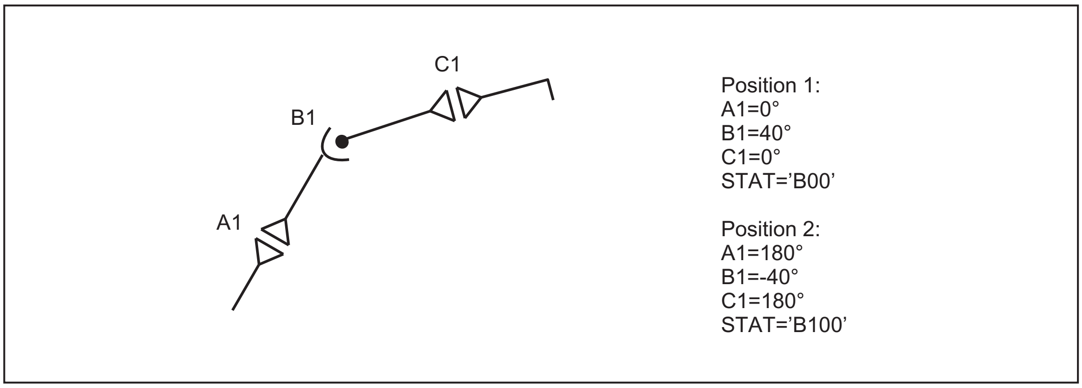


Figure 6-19 Ambiguity of axis B1

6.6.4 Example of ambiguity in rotary axis position

The rotary axis position shown in the following diagram can be approached in the negative or positive direction. The direction is programmed under address A1.

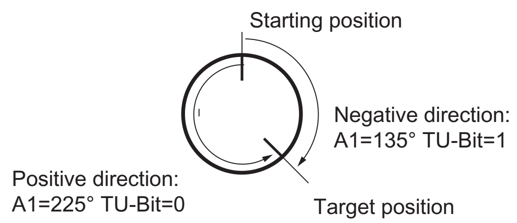


Figure 6-20 Ambiguity in rotary axis position

6.6.5 PTP/CP switchover in JOG mode

In JOG mode, the transformation can be switched on and off via a channel-specific NC/PLC interface signal. This control signal is only effective in JOG mode when the transformation is active.

After returning to AUTO mode, the state which was last active before switchover is restored.

NC/PLC interface signals

- Request to switch the traversing type:
  DB21, ... DBX29.4 (activate PTP travel)
- Feedback from the active traversing type:
  DB21, ... DBX317.6 (PTP travel active)

Operating mode change

The "Cartesian PTP travel" function is only useful in the AUTO and MDI modes of operation. The `CP` setting is automatically activated if the operating mode is switched to JOG. If the mode is then switched back to AUTO or MDI, the mode that was last active in either mode is made active again.

REPOS

The setting for "Cartesian PTP travel" is not altered during re-positioning. If `PTP` was set in the interruption block, then repositioning also takes place with `PTP`. For an inclined axis "TRAANG", only CP travel is active in `REPOS` mode.

6.7 Cartesian manual travel (optional)

Note

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

Function

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

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

Adjustment and activation is done using machine data:

MD21106 $MC_CART_JOG_SYSTEM (coordinate systems for Cartesian JOG)

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

Note

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

Reference:
Function Manual Basic Functions; Axes, Coordinate Systems, Frames (K2)

Representation of the reference system in the coordinate system:

	**WCS** = Workpiece zero		**TCS** = Tool reference point

Selecting reference systems

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

SD42650 $SC_CART_JOG_MODE

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

Translation

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

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

Translation in the BCS

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

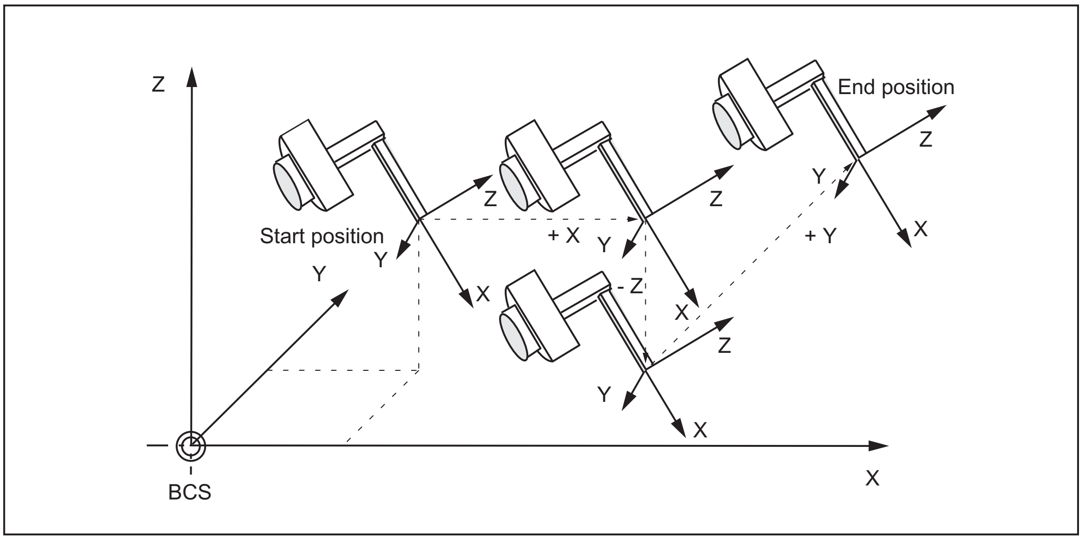


Figure 6-21 Cartesian manual travel in the basic coordinate system (translation)

Translation in the WCS

The workpiece coordinate system (WCS) lies in the workpiece zero. The workpiece coordinate system can be shifted and rotated relative to the reference system via frames. As long as the frame rotation is active, the traversing movements correspond to the translation of the movements in the basic coordinate system.

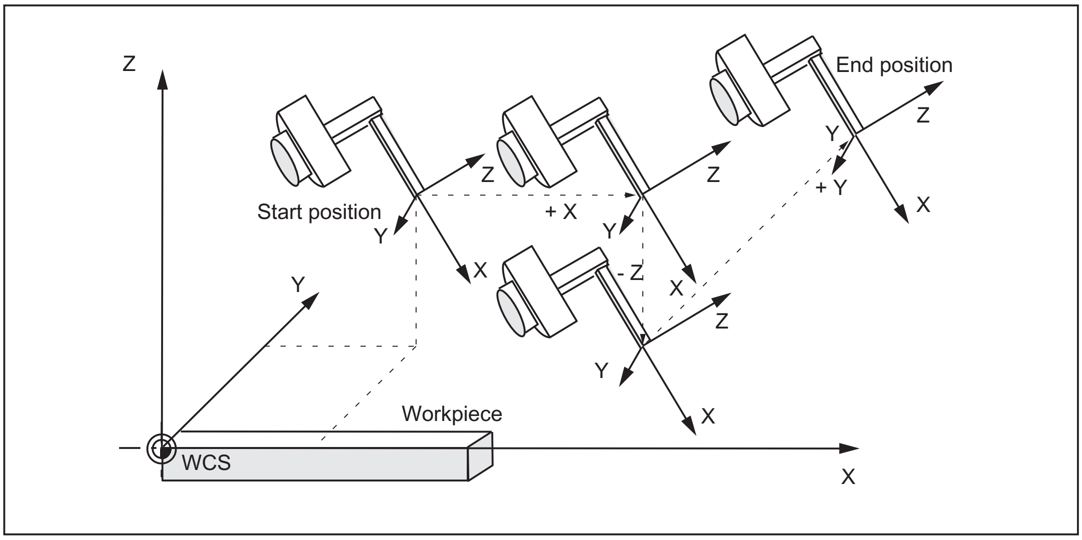


Figure 6-22 Cartesian manual travel in the workpiece coordinate system (translation)

Translation in the TCS

The tool coordinate system (TCS) lies in the tool tip. Its direction depends on the current setting of the machine, since the tool coordinate system moves during the motion.

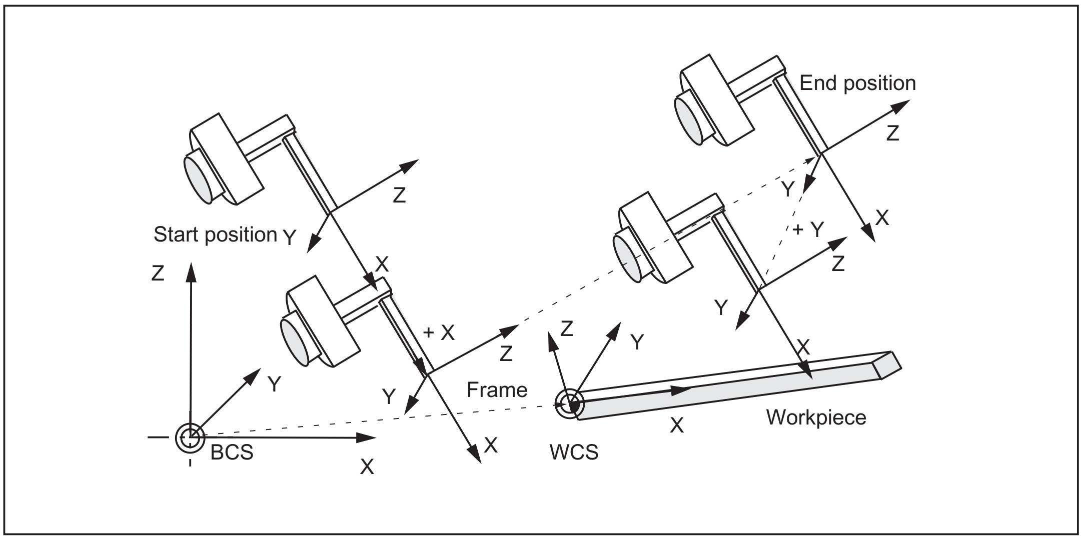


Figure 6-23 Cartesian manual travel in the tool coordinate system (translation)

Translation and orientation in the TCS simultaneously

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

Orientation

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

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

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

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

Programming rotations:

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

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

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

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

Orientation in WCS

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

Orientation in BCS

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

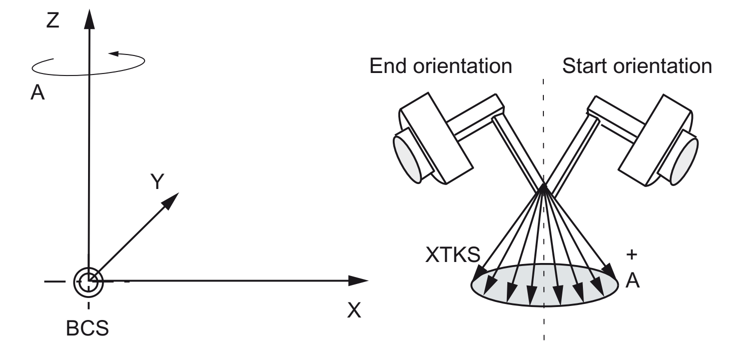


Figure 6-24 Cartesian manual travel in the basic coordinate system, orientation angle A

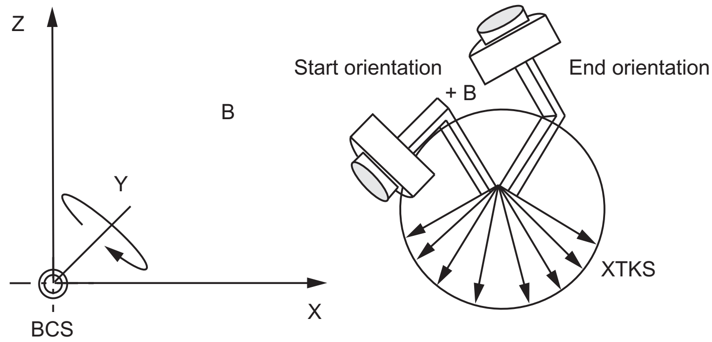


Figure 6-25 Cartesian manual travel in the basic coordinate system, orientation angle B

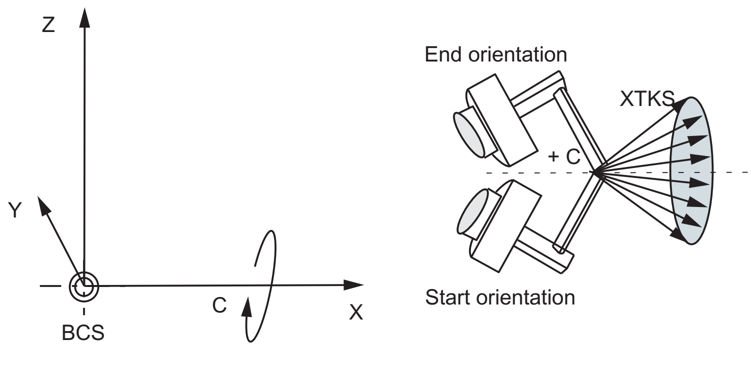


Figure 6-26 Cartesian manual travel in the basic coordinate system, orientation angle C

Orientation in TCS

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

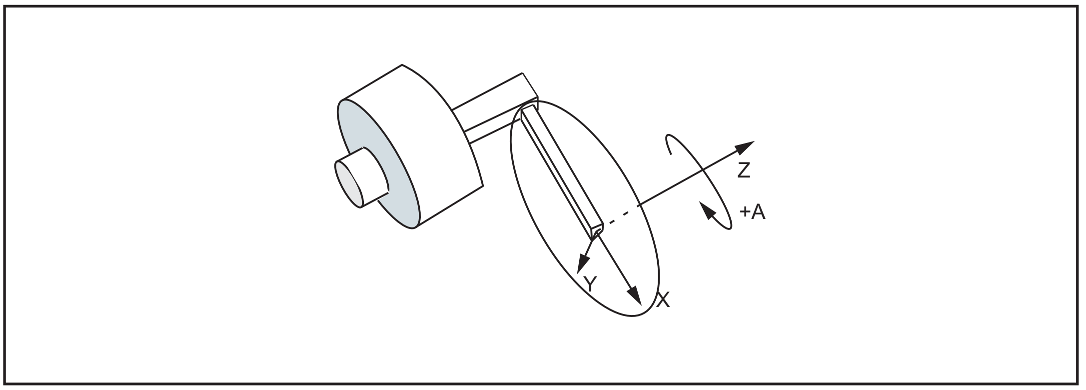


Figure 6-27 Cartesian manual travel in the tool coordinate system, orientation angle A

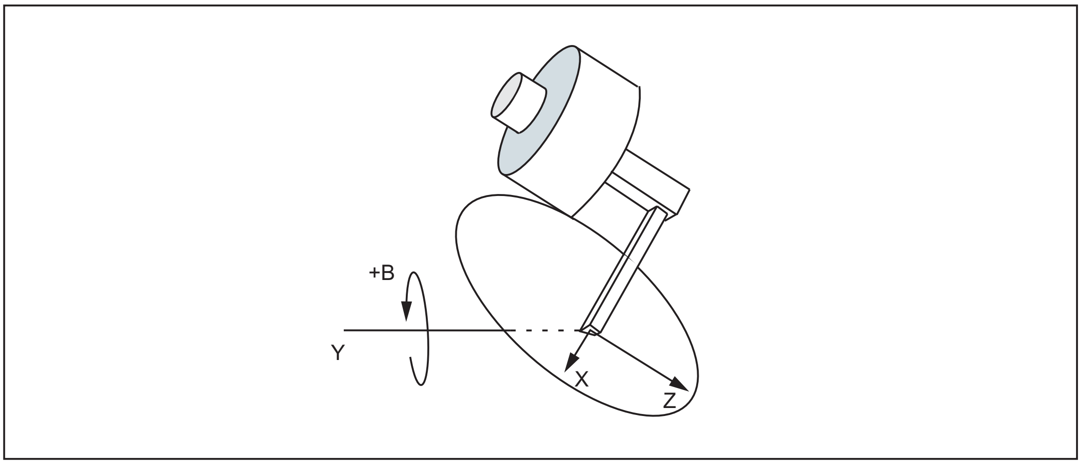


Figure 6-28 Cartesian manual travel in the tool coordinate system, orientation angle B

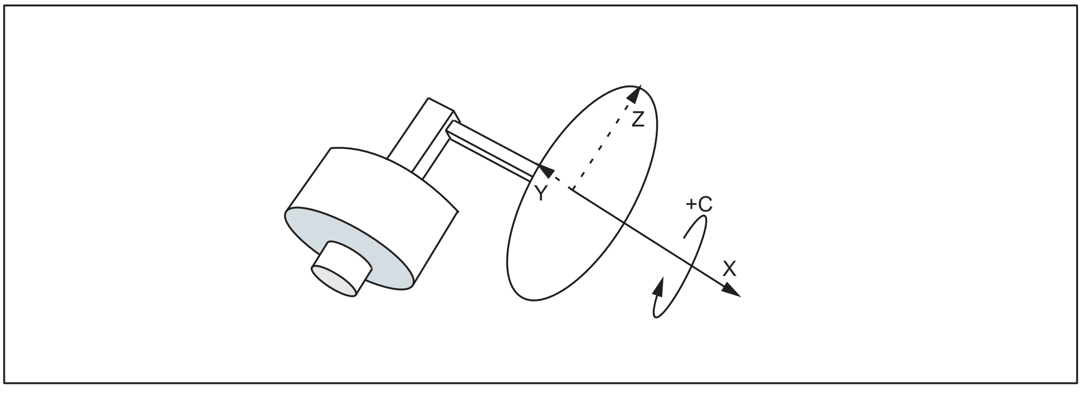


Figure 6-29 Cartesian manual travel in the tool coordinate system, orientation angle C

Supplementary conditions

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

The following supplementary conditions must be observed:

- "Handling transformation package" option with 5-axis or 6-axis transformation is set
- Virtual orientation axes must be defined via the following machine data:

  MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1[n]
- DB21, ... DBX29.4 == 0 (activate PTP travel)
- MD21106 $MC_CART_JOG_SYSTEM > 0

Table 6- 2 Conditions for Cartesian manual travel

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

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

Activation

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

- The Cartesian manual travel function is activated with the following machine data:

  MD21106 $MC_CART_JOG_SYSTEM > 0

  The BCS, WCS or TCS reference systems are enabled via MD 21106 $MC_CART_JOG_SYSTEM.
- JOG traverse motion via SD42650 SC_CART_JOG_MODE

  Standard behavior as before: Bits 0 to 2 = 0, bits 8 to 10 = 0.

  Reference system for translation via bits 0-2 and the reference system for orientation via bits 8-10.

  If not all of the bits are set to 0, the process uses the new function. The reference systems for translation and orientation may be set independently.

SD42650 $SC_CART_JOG_MODE (only set one bit):

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

Combining reference systems

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

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

6.8 Activating transformation machine data via part program/softkey

6.8.1 Functionality

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

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

MD24100 $MC_TRAFO_TYPE_1

up to

MD24460 $MC_TRAFO_TYPE_10.

Characteristics

Transformation machine data are NEWCONFIG effective.

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

Provided that no transformation is selected (activated) when a `NEWCONF` command is issued (regardless whether via the NEWCONF NC program command, the HMI or implicitly following Reset or end of program), the machine data listed above can be altered without restriction and then activated.

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

6.8.2 Constraints

Change machine data

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

These are generally all machine data assigned to a transformation via the associated transformation data group. Machine data that are included in the group of an active transformation, but not in use, can be altered (although this would hardly be meaningful). For example, it would be possible to change machine data MD24564 $MC_TRAFO5_NUTATOR_AX_ANGLE_n for an active transformation with MD24100 $MC_$MC_TRAFO_TYPE = 16 (5-axis transformation with rotatable tool and two mutually perpendicular rotary axes A and B) since this particular machine data is not involved in the transformation.

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

Note

In the case of a program interruption (Repos, deletion of distance to go, ASUBs, etc.), the control system requires a number of different blocks that have already been executed for the repositioning operation. The rule forbidding the machine data of an active transformation to be altered also refers to these blocks.

Example:

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

Assume that the second transformation is active when the NEWCONFIG command is executed. In this case, all machine data that relate only to the first transformation may be changed, e.g.:

MD24500 $MC_TRAFO5_PART_OFFSET_1

but not, for instance:

MD24650 $MC_TRAFO5_BASE_TOOL_2

or

MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE

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

Defining geometry axes

Geometry axes must be defined **before** starting the control system with the following machine data:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_X[n]

or

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[n]

Changing the assignment

The assignment of a transformation data set to a transformation is determined by the sequence of entries in MD24100 $MC_TRAFO_TYPE_X. The first entry in the table is assigned to the first transformation data set, and accordingly the second entry to the second data set. This assignment may (and can) not be altered for an active transformation.

Example:

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

MD24100 $MC_TRAFO_TYPE_1 = 16

; orientation transformation, 1st orientation trafo data set

MD24200 $MC_TRAFO_TYPE_2 = 256 : Transmit transformations

MD24300 $MC_TRAFO_TYPE_3 = 18

; orientation transformation, 2nd orientation trafo data set

The first data set for orientation transformations is assigned to the first transformation (equaling the first orientation transformation) and the second transformation data set to the third transformation (equaling the second orientation transformation).

If the third transformation is active when the `NEWCONFIG` command is executed, it is not permissible to change the first transformation into a transformation of another group (e.g. TRACYL) since, in this case, the third transformation would then not become the second orientation transformation, but the first.

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

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

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

MD20110 $MC_RESET_MODE_MASK

MD20112 $MC_START_MODE_MASK

and

MD20140 $MC_TRAFO_RESET_VALUE

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

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

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

6.8.4 List of machine data affected

Machine data which can be made NEWCONFIG compatible are listed below.

All transformations

Machine data which are relevant for all transformations:

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

Orientation transformations

Machine data which are relevant for orientation transformations:

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

Transmit transformations

Machine data which are relevant for Transmit transformations:

- MD24920 $MC_TRANSMIT_BASE_TOOL_1 and
  MD24970 $MC_TRANSMIT_BASE_TOOL_2
- MD24900 $MC_TRANSMIT_ROT_AX_OFFSET_1 and
  MD24950 $MC_TRANSMIT_ROT_AX_OFFSET_2

- MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1 and
  MD24960 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_2
- MD24911 MC_RANSMIT_POLE_SIDE_FIX_1 and
  MD24961 $MC_TRANSMIT_POLE_SIDE_FIX_2

Tracyl transformations

Machine data which are relevant for Tracyl transformations:

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

Inclined axis transformations

Machine data which are relevant for inclined axis transformations:

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

Chained transformations

Machine data which are relevant for chained transformations:

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

Persistent transformation

Machine data which are relevant for persistent transformations:

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

Not transformation-specific

Machine data that are not transformation-specific. they are not uniquely assigned to a particular transformation data set or they are relevant even when a transformation is not active:

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

6.9 Constraints

6.9.1 Chained transformations

Two transformations can be chained.

However, not just any transformation can be chained to another one.

In this case, the following restrictions apply:

- The **first** transformation of the chain has to be one of the following transformations:
  - Orientation transformation (3-axis, 4-axis, 5-axis transformations, universal milling head).
  - Transmit
  - Surface line transformation
  - Inclined axis
- The **second** transformation must be an**inclined** axis transformation.
- Only two transformations may be chained.

It is allowed (for test purposes, for instance), to enter only a single transformation into the chaining list.

6.10 Examples

6.10.1 TRANSMIT

The following example relates to the configuration illustrated in "Figure 6-30 Groove with groove wall offset, cylinder coordinates (Page 423)" and shows the sequence of main steps required to configure the axes and activate TRANSMIT.

```
; General axis configuration for rotation
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"          ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1]="Y"          ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"          ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1        ; X as channel axis 1
MD20060 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0        ; Y no channel axis
MD20060 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 2        ; Z as channel axis 2
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0]="XC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1]="ZC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2]="CC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="ASC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4] = " "
MD20070 $MC_AXCONF_MACHAX_USED[0] = 2             ; XC as machine axis 2
MD20070 $MC_AXCONF_MACHAX_USED[1]=3               ; ZC as machine axis 3
MD20070 $MC_AXCONF_MACHAX_USED[2]=1               ; CC as machine axis 1
MD20070 $MC_AXCONF_MACHAX_USED[3] = 4             ; ASC as machine axis 4
MD20070 $MC_AXCONF_MACHAX_USED[3] = 0             ; empty
MD20070 $MA_SPIND_ASSIGN_TO_MACHAX[AX1]= 1        ; C is spindle 1
MD20070 $MA_SPIND_ASSIGN_TO_MACHAX[AX2]= 0        ; X is no spindle
MD20070 $MA_SPIND_ASSIGN_TO_MACHAX[AX3]= 0        ; Z is no spindle
MD20070 $MA_SPIND_ASSIGN_TO_MACHAX[AX4]= 2        ; AS is spindle 2
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]="CM"        ; 1. Machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]="XM"        ; 2. Machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]="ZM"        ; 3. Machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="ASM"       ; 4. Machine axis
```

; prepare for TRANSMIT (as first and only transformation)	
$MA_ROT_IS_MODULO[3] = TRUE	; c as modulo axis
MD24100 $MC_TRAFO_TYPE_1 = 256	;TRANSMIT transformation
MD24110 $MC_TRAFO_AXES_IN_1[0] = 1	; channel axis perpendicular to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[1] = 3	; channel rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[2]=2	; channel axis parallel to rotary axis
MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] =1	; 1. channel axis becomes GEOAX X
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=3	; 2. channel axis becomes GEOAX Y
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=2	; 3. channel axis becomes GEOAX Z
MD24900 $MC_TRANSMIT_ROT_AX_OFFSET_1=0	; rotation position X-Y plane against zero position of the rotary axis
MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1=FALSE	; Rotary axes turns
MD24920 $MC_TRANSMIT_BASE_TOOL_1 [0]=0.0	; tool center distance in X
MD24920 $MC_TRANSMIT_BASE_TOOL_1 [1]=0.0	; tool center distance in Y
MD24920 $MC_TRANSMIT_BASE_TOOL_1 [2]=0.0	; tool center distance in Z
; activation TRANSMIT	
; Programming in X,Y,Z	
; Return to rotational operation	
TRAFOOF	

See also

TRACYL (Page 420)

6.10.2 TRACYL

The following figure shows an example relating to the configuration of axes and shows the sequence of main steps required to configure the axes up to activation by TRACYL.

; General axis configuration for rotation	
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"	; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1]="Y"	; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"	; Geometry axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1	; X as channel axis 1
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2	; Y no channel axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3	; Z as channel axis 2

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0]="XC"	
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1]="YC"	
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2]="ZC"	
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="CC"	
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4]="ASC"	
MD20070 $MC_AXCONF_MACHAX_USED[0] = 2	; X as machine axis 2
MD20070 $MC_AXCONF_MACHAX_USED[1] = 3	; Y as machine axis 3
MD20070 $MC_AXCONF_MACHAX_USED[2] = 4	; Z as machine axis 4
MD20070 $MC_AXCONF_MACHAX_USED[3] = 1	; C as machine axis 1
MD20070 $MC_AXCONF_MACHAX_USED[4] = 5	; AS as machine axis 5
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX1]= 1	; C is spindle 1
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX2]= 0	; X is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX3]= 0	; Y is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX4]= 0	; Z is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX5]= 2	; AS is spindle 2
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]="CM"	; 1. machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]="XM"	; 2. machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]="YM"	; 3. machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="ZM"	; 4. machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4]="ASM"	; 5. machine axis

; prepare for TRACYL (as first and only transformation)

MD24100 $MC_TRAFO_TYPE_1 = 513	; Transformation TRACYL with groove wall offset
MD24110 $MC_TRAFO_AXES_IN_1[0] = 1	; channel axis radial to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[1] = 4	; Channel axis in generated cylinder surface perpendicular to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3	; channel axis parallel to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[3] = 2	; Channel axis special axis to index [0]
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1	; 1. channel axis becomes GEOAX X
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 4	; 2. channel axis becomes GEOAX Y
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3	; 3. channel axis becomes GEOAX Z
MD24800 $MC_TRACYL_ROT_AX_OFFSET_1 = 0	; rotation position X-Y plane against zero position of the rotary axis

MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_1 = FALSE	; Rotary axes turns
MD24820 $MC_TRACYL_BASE_TOOL_1 [0] = 0.0	; tool center distance in X
MD24820 $MC_TRACYL_BASE_TOOL_1 [1] = 0.0	; tool center distance in Y
MD24820 $MC_TRACYL_BASE_TOOL_1 [2] = 0.0	; tool center distance in Z

; activation TRACYL(40.0)

; Programming in Y and Z, see the following example

; Return to rotational operation

TRAFOOF

Programming with groove wall offset

(TRAFO_TYPE_n=513)

Contour

It is possible to produce a groove which is wider than the tool by using address OFFN to program the compensation direction (G41, G42) in relation to the programmed reference contour and the distance of the groove side wall from the reference contour (see following figure).

Tool radius

The tool radius is automatically taken into account with respect to the groove side wall (see simplified figure). The full functionality of the plane tool radius compensation is available (steady transition at outer and inner corners as well as solution of bottleneck problems).

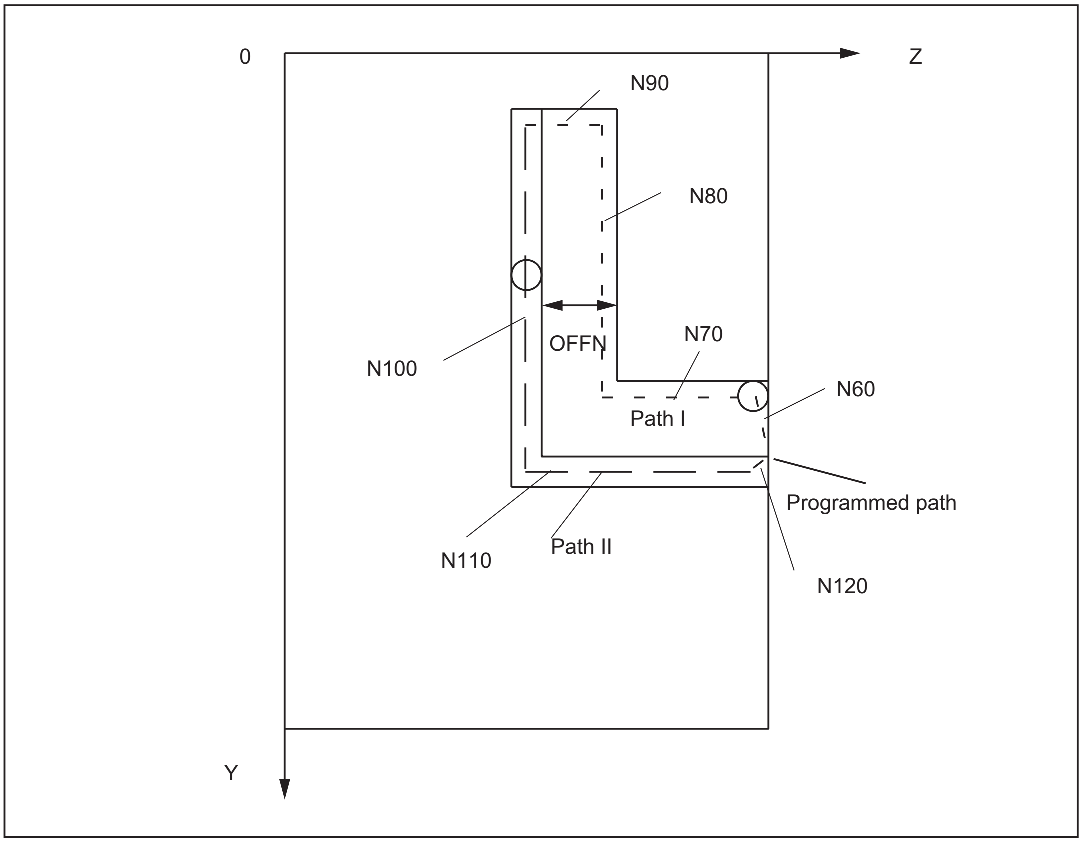


Figure 6-30 Groove with groove wall offset, cylinder coordinates

Example program that which guides the tool after transformation selection on path I via path II back to the starting position

Program code	Comment
N1 SPOS=0;	; Transfer the spindle to rotary axis operation
N5 G0 X25 Y0 Z105 CC=200 F5000 G64	; Positioning of the machine above groove center
N10 TRACYL(40.)	; Transformation selection with reference diameter: 40 mm
N20 G19 G90	; Machining plane is cylinder surface
N30 T1 D1	; Tool selection, can also be before TRACY (..)
N30 T1 D1	; Infeed tool to groove base
N50 OFFN=12	; Determine groove wall distance; it does need not to be in its own line

Program code	Comment
; Approach of groove wall	
N60 G1 Z100 G42	; TRC selection to approach groove wall
Machining groove sector path I	
N70 G1 Z50	; Groove part parallel to the cylinder plane
N80 G1 Y10	; Groove part parallel to the circumference
; Approaching groove wall for path II	
N90 OFFN=4 G42	; Specify groove wall distance and TRC selection to approach the groove wall
; Machining groove sector path II	
N100 G1 Y70	; Corresponds to CC=200 degrees
N110 G1 Z100	; Revert to initial value
;Retract from groove wall	
N120 G1 Z105 G40	; TRC deselection to retract from the groove wall
N130 G0 X25	; Retract from the groove
N140 TRAFOOF N150 G0 X25 Y0 Z105 CC=200 D0	; Return to the starting point and deselect the tool offset

Programming without groove wall offset

TRACYL without groove wall offset with supplementary linear axis (TRAFO_TYPE_n = 514)

; For the following part program the following machine data settings are a prerequisite:

MD20070 $MC_AXCONF_MACHAX_USED[0]=1	; X as machine axis 1
MD20070 $MC_AXCONF_MACHAX_USED[1] = 2	; Y as machine axis 2
MD20070 $MC_AXCONF_MACHAX_USED[2] = 3	; Z as machine axis 3
MD20070 $MC_AXCONF_MACHAX_USED[3] = 4	; C as machine axis 4
MD20070 $MC_AXCONF_CHANAX_NAME_TAB[1] = "Y2"	
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1	; X as channel axis 1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 2	; Y no channel axis
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3	; Z as channel axis 2
MD24100 $MC_TRAFO_TYPE_1 = 514	; TRACYL without groove wall offset only with tool length offset
MD24110 $MC_TRAFO_AXES_IN_1[0] = 1	; channel axis radial to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[1] = 4	; Channel axis in cylinder surface
	; perpendicular to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3	; channel axis parallel to rotary axis

MD24110 $MC_TRAFO_AXES_IN_1[3] = 2	; Channel axis special axis to index [0]
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1	; 1. channel axis becomes GEOAX X
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 4	; 2. channel axis becomes GEOAX Y
MD24110 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3	; 3. channel axis becomes GEOAX Z
MD24808 $MC_TRACYL DEFAULT MODE_1 =0	; or not set at all

Tool data:

$TC_DP1[1,1]=120	; Tool type shaft miller
$TC_DP2[1,1] = 0	
$TC_DP3[1,1]=0	; Length offset vector
$TC_DP4[1,1]=25	
$TC_DP5[1.1] =5	
$TC_DP6[1.1] =4	; Radius, tool radius

Part program:

Program code	**Comment**
N1001 T1 D1 G54 G19 G90 F5000 G64	; Selection of the 1st TRACYL **without** groove wall offset
N1005 G0 X25 Y0 Z105 A=200	
N1010 TRACYL(40.)	; Transformation selection
N1040 G1 X20	
N1060 G1 Z100	
N1070 G1 Z50	
N1080 G1 Y10	
N1140 TROFOOF	
N1150 G0 X25 Y0 Z105 A=200	; Selection of the 1st TRACYL **with** groove wall offset
N2010 G0 TRACYL(40.,1,1)	; TRACYL (40., ,1) would also be possible
N2040 G1 X20	
N2060 G1 Z100	
N2070 G1 Z50	
N2080 G1 Y10	
N2140 TROFOOF	

6.10.3 TRAANG

For the configuration shown in Figure "Groove with Groove Wall Offset, Cylinder Coordinates", an example relating to the configuration of axes which shows the sequence of main steps required to configure the axes up to activation by TRAANG is shown.

```
; General axis configuration for grinding
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"              ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = " "            ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"              ; Geometry axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 0            ; X no channel axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0            ; Y no channel axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 1            ; Z as channel axis 1
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "Z"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "C"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "AS1"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "MU"
MD20070 $MC_AXCONF_MACHAX_USED[0] = 3                 ; Z as machine axis 3
MD20070 $MC_AXCONF_MACHAX_USED[1]=1                   ; C as machine axis 1
MD20070 $MC_AXCONF_MACHAX_USED[2] = 4                 ; AS as machine axis 4
MD20070 $MC_AXCONF_MACHAX_USED[3] = 2                 ; MU as machine axis 2
MD20070 $MC_AXCONF_MACHAX_USED[3] = 0                 ; empty
MD20070 $MC_AXCONF_MACHAX_USED[3] = 0                 ; empty
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX1]= 1            ; C is spindle 1
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX2]= 0            ; X is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX3]= 0            ; Z is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX4]= 2            ; AS is spindle 2
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]= "C1"           ; 1. Machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]= "MU"           ; 2. Machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]= "MZ"           ; 3. Machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="AS1"           ; 4. Machine axis
```

```
; prepare for TRAANG (as first and only transformation)
MD24100 $MC_TRAFO_TYPE_1 = 1024                       ; Transformation TRAANG
MD24110 $MC_TRAFO_AXES_IN_1[0]=4                      ; Channel axis inclined axis
MD24110 $MC_TRAFO_AXES_IN_1[1]=1                      ; channel axis parallel to axis Z
MD24110 $MC_TRAFO_AXES_IN_1[2] = 0                    ; Channel axis not active
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4             ; X 1st channel axis
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 0          ; Y 2nd channel axis
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 1          ; Z 3rd channel axis
```

```
MD24700 $MC_TRAANG_ANGLE_1 = 30.             ; Angle of inclined axis
MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0       ; tool center distance in X
MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0       ; tool center distance in Y
MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0       ; tool center distance in Z
TRAANG                                       ; activation
                                             ; Programming in X,Y,Z
TRAFOOF                                      ; Return to rotational operation
```

6.10.4 Chained transformations

Examples

The following section determines:

- The general channel configuration
- Single transformations
- Chained transformations consisting of previously defined single transformations
- Activation of single transformations
- Activation of chained transformations

The examples include the following transformations:

- 5-axis transformation with rotatable tool and axis sequence AB (trafo type 16)
- Transmit (trafo type 256)
- Inclined axis (trafo type 1024)
- Chaining of the 1st and 3rd transformation (trafo type 8192)
- Chaining of the 2nd and 3rd transformation (trafo type 8192)

General channel configuration

```
CHANDATA(1)                               ; Channel data in channel 1
MD20070 $MC_AXCONF_MACHAX_USED[0]=1
MD20070 $MC_AXCONF_MACHAX_USED[1]=2
MD20070 $MC_AXCONF_MACHAX_USED[2] = 3
MD20070 $MC_AXCONF_MACHAX_USED[3] = 4
MD20070 $MC_AXCONF_MACHAX_USED[4]=5
MD20070 $MC_AXCONF_MACHAX_USED[5]=6
MD20070 $MC_AXCONF_MACHAX_USED[6]=7
MD20070 $MC_AXCONF_MACHAX_USED[7] = 0
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="A"
```

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

Single transformations

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


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

```
; 3. TRAANG
MD24300 $MC_TRAFO_TYPE_3 = 1024         ; TRAANG
MD24310 $MC_TRAFO_AXES_IN_3[0] = 1
MD24310 $MC_TRAFO_AXES_IN_3[1] = 3
MD24310 $MC_TRAFO_AXES_IN_3[2] = 2
MD24310 $MC_TRAFO_AXES_IN_3[3] = 0
MD24310 $MC_TRAFO_AXES_IN_3[4] = 0
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =1
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =3
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =2
MD24700 $MC_TRAANG_ANGLE_1 = 45.
MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0.0
```

Chained transformations

```
; 4. TRACON (Chaining TRAORI/TRAANG)
MD24400 $MC_TRAFO_TYPE_4 = 8192
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =2
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3
MD24995 $MC_TRACON_CHAIN_1[0] = 1
MD24995 $MC_TRACON_CHAIN_1[1] = 3
MD24995 $MC_TRACON_CHAIN_1[2] = 0
; 5. TRACON (Chaining TRANSMIT/TRAANG)
MD24430 $MC_TRAFO_TYPE_5 = 8192
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =6
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3
MD24996 $MC_TRACON_CHAIN_2[0] = 2
MD24996 $MC_TRACON_CHAIN_2[1] = 3
MD24996 $MC_TRACON_CHAIN_2[2] = 0
```

Part program (extracts)

Example of an NC program which uses the set transformations:

Program code	Comment
; Call single transformations	
	; Tool specification
$TC_DP1[1,1]=120	; Tool type
$TC_DP3[1,1] = 10	; Tool length
n2 x0 y0 z0 a0 b0 f20000 t1 d1n4 x20	
n30 TRANSMIT	; Switch on TRANSMIT
n40 x0 y20	
n50 x-20 y0	
n60 x0 y-20	
n70 x20 y0	
n80 TRAFOOF	; Switch off TRANSMIT
n130 TRACYL(45.)	; Activate inclined axis transformation, parameters: Angle 45°
n140 x0 y0 z20	
n150 x-20 z0	
n160 x0 z-20	
n170 x20 z0	

Note

The above examples assume that the angle of the "inclined axis" can be set on the machine and is set to 0° when the single transformation is activated.

Program code	Comment
; 1. Activate chained transformations	
; TRAORI + TRAANG	
n230 TRACON(1, 45.)	; 1. of the 2 chained transformations to be switched on
	; The previously active transformation TRAANG is automatically deselected
	; The parameter for the inclined axis is 45°
n240 x10 y0 z0 a3=-1 C3 =1 oriwks	
n250 x10 y20 b3 = 1 c3 = 1	

Program code	Comment
; 2. Activate chained transformations	
; TRANSMIT + TRAANG	
n330 TRACON(2, 40.)	; 2. activate chained transformation
	; The parameter for the inclined axis is 40°
n335 x20 y0 z0	
n340 x0 y20 z10	
n350 x-20 y0 z0	
n360 x0 y-20 z0	
n370 x20 y0 z0	
n380 TRAFOOF	2. deactivate chained transformation
n1000 M30	

6.10.5 Activating transformation MD via a part program

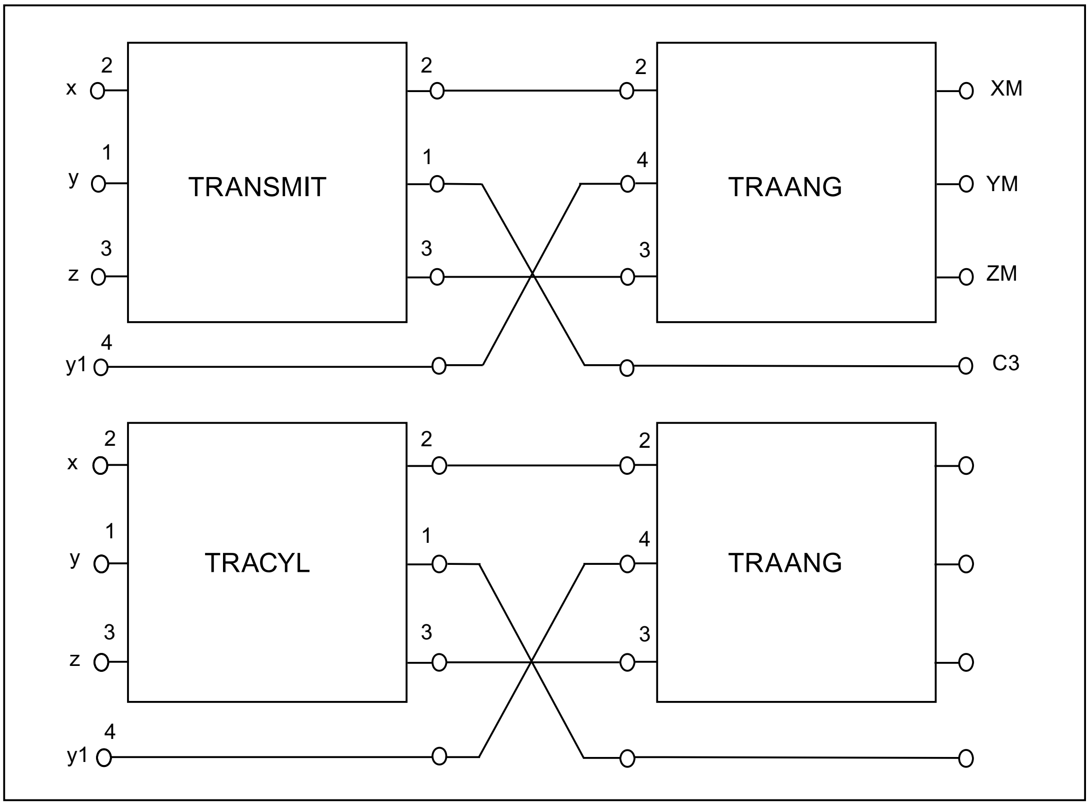
It would be permissible in the following example to reconfigure (write) a machine data affecting the second transformation (e.g. MD24650 $MC_TRAFO5_BASE_TOOL_2[2]) in block N90, since writing a machine data alone does not activate it. However, if the program remained otherwise unchanged, an alarm would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

Example program:

Program code	Comment
N40 TRAORI(2)	; Select 2nd orientation transformation
N50 X0 Y0 Z0 F20000 T1 T1	
N60 A50 B50	
N70 A0 B0	
N80 X10	
N90 $MC_TRAFO5_BASE_TOOL_1[2] = 50	; Overwrite a machine data item of the 1st orientation transformation
N100 A20	
N110 X20	
N120 X0	
N130 NEWCONF	; Accept new machine data
N140 TRAORI(1)	; Selection of the 1st orientation transformation MD is effective
N150 G19 X0 Y0 Z0	
N160 A50 B50	
N170 A0 B0	
N180 TRAFOOF	
N190 M30	

6.10.6 Axis positions in the transformation chain

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



Machine data

CHANDATA(1)

MD24100 $MC_TRAFO_TYPE_1=256 ; TRANSMIT
MD24110 $MC_TRAFO_AXES_IN_1[0] = 2
MD24110 $MC_TRAFO_AXES_IN_1[1] = 1
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 2
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3

MD24200 $MC_TRAFO_TYPE_2=512 ; TRACYL
MD24210 $MC_TRAFO_AXES_IN_2[0]=2
MD24210 $MC_TRAFO_AXES_IN_2[1]=1

MD24210 $MC_TRAFO_AXES_IN_2[2]=3
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] =2
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] =1
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] =3

MD24300 $MC_TRAFO_TYPE_3=1024 ; TRAANG
MD24310 $MC_TRAFO_AXES_IN_3[0] = 2
MD24310 $MC_TRAFO_AXES_IN_3[1]=4
MD24310 $MC_TRAFO_AXES_IN_3[2] = 3
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =2
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =4
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =3
MD24700 $MC_TRAANG_ANGLE_1 = 45.
MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0.0

1st TRANSMIT / TRAANG chaining
MD24400 $MC_TRAFO_TYPE_4=8192 ; TRACON (1)
MD24995 $MC_TRACON_CHAIN_1[0] = 1
MD24995 $MC_TRACON_CHAIN_1[1] = 3
MD24995 $MC_TRACON_CHAIN_1[2] = 0
MD24995 $MC_TRACON_CHAIN_1[3] = 0
MD24410 $MC_TRAFO_AXES_IN_4[0]=1
MD24410 $MC_TRAFO_AXES_IN_4[1]=2
MD24410 $MC_TRAFO_AXES_IN_4[2]=3
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =2
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3

2nd TRACYL / TRAANG chaining
MD24430 $MC_TRAFO_TYPE_5=8192 ; TRACON (2)
MD24996 $MC_TRACON_CHAIN_2[0] = 2
MD24996 $MC_TRACON_CHAIN_2[1] = 3
MD24996 $MC_TRACON_CHAIN_2[2]=0
MD24996 $MC_TRACON_CHAIN_2[3]=0
MD24432 $MC_TRAFO_AXES_IN_5[0]=1
MD24432 $MC_TRAFO_AXES_IN_5[1]=2
MD24432 $MC_TRAFO_AXES_IN_5[2]=3

MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =2
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3

M17

Part program

Program code	Comment
N10 $TC_DP1[1,1]=120	
N20 $TC_DP3[1,1]= 20	
N30 $TC_DP4[1,1]=0	
N40 $TC_DP5[1,1]=0	
N50	
N60 X0 Y0 Z0 F20000 T1 D1	
N70	
N80	; cyclical reading of the variables in the synchronous action
N90 ID=1 WHENEVER TRUE DO $R0=$AA_ITR[X,0] $R1=$AA_ITR[X,1] $R2=$AA_ITR[X,2]	
N100 ID=2 WHENEVER TRUE DO $R3=$AA_IBC[X] $R4=$AA_IBC[Y] $R5=$AA_IBC[Z]	
N110 ID=3 WHENEVER TRUE DO $R6=$VA_IW[X]-$AA_IW[X]	
N120 ID=4 WHENEVER TRUE DO $R7=$VA_IB[X]-$AA_IB[X]	
N130 ID=5 WHENEVER TRUE DO $R8=$VA_IBC[X]-$AA_IBC[X]	
N140 ID=6 WHENEVER TRUE DO $R9=$VA_ITR[X,1]-$AA_ITR[X,1]	
N150	
N160	; 1. TRANSMIT / TRAANG chaining
N170 TRACON(1,)	
N180 X20 Y0 Z0	
N190 X0 Y20 Z10	
N200 X-20 Y0 Z0	
N210 X0 Y-20 Z0	
N220 X20 Y0 Z0	
N230 TRAFOOF	
N240	
N250	; 2. TRACYL/ TRAANG chaining
N260 TRACON (2, 40.)	
N270 X20 Y0 Z0	
N280 X0 Y20 Z10	
N290 X-20 Y0 Z0	
N300 X0 Y-20 Z0	
N310 X20 Y0 Z0	
N320 TRAFOOF	
N330	
N340 M30	

6.11 Data lists

6.11.1 Machine data

6.11.1.1 TRANSMIT

Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20140	TRAFO_RESET_VALUE	Basic transformation position
22534	TRAFO_CHANGE_M_CODE	M code for transformation changeover
24100	TRAFO_TYPE_1	Definition of the 1st transformation in channel
24110	TRAFO_AXES_IN_1	Axis assignment for the 1st transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Geo-axis assignment for 1st transformation
24200	TRAFO_TYPE_2	Definition of the 2nd transformation in channel
24210	TRAFO_AXES_IN_2	Axis assignment for the 2nd transformation
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Geo-axis assignment for 2nd transformation
24300	TRAFO_TYPE_3	Definition of the 3rd transformation in channel
24310	TRAFO_AXES_IN_3	Axis assignment for the 3rd transformation
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Geo-axis assignment for 3rd transformation
24400	TRAFO_TYPE_4	Definition of the 4th transformation in channel
24410	TRAFO_AXES_IN_4	Axis assignment for the 4th transformation
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Geo-axis assignment for 4th transformation
24430	TRAFO_TYPE_5	Definition of the 5th transformation in channel
24432	TRAFO_AXES_IN_5	Axis assignment for the 5th transformation
24434	TRAFO_GEOAX_ASSIGN_TAB_5	Geo-axis assignment for 5th transformation
24440	TRAFO_TYPE_6	Definition of the 6th transformation in channel
24442	TRAFO_AXES_IN_6	Axis assignment for the 6th transformation
24444	TRAFO_GEOAX_ASSIGN_TAB_6	Geo-axis assignment for 6th transformation
24450	TRAFO_TYPE_7	Definition of the 7th transformation in channel
24452	TRAFO_AXES_IN_7	Axis assignment for the 7th transformation
24454	TRAFO_GEOAX_ASSIGN_TAB_7	Geo-axis assignment for 7th transformation
24460	TRAFO_TYPE_8	Definition of the 8th transformation in channel
24462	TRAFO_AXES_IN_8	Axis assignment for the 8th transformation
24464	TRAFO_GEOAX_ASSIGN_TAB_8	Geo-axis assignment for 8th transformation
24900	TRANSMIT_ROT_AX_OFFSET_1	Deviation of rotary axis from zero position in degrees (1st TRANSMIT)

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

6.11.1.2 TRACYL

Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20140	TRAFO_RESET_VALUE	Basic transformation position
20144	TRAFO_MODE_MASK	Selection of the kinematic transformation function
24100	TRAFO_TYPE_1	Definition of the 1st transformation in channel
24110	TRAFO_AXES_IN_1	Axis assignment for the 1st transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Geo-axis assignment for 1st transformation
24130	TRAFO_INCLUDES_TOOL_1	Tool handling with active transformation 1.
24200	TRAFO_TYPE_2	Definition of the 2nd transformation in channel
24210	TRAFO_AXES_IN_2	Axis assignment for the 2nd transformation
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Geo-axis assignment for 2nd transformation
24230	TRAFO_INCLUDES_TOOL_2	Tool handling with active transformation 2.
24300	TRAFO_TYPE_3	Definition of the 3rd transformation in channel
24310	TRAFO_AXES_IN_3	Axis assignment for the 3rd transformation
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Geo-axis assignment for 3rd transformation
24330	TRAFO_INCLUDES_TOOL_3	Tool handling with active transformation 3.
24400	TRAFO_TYPE_4	Definition of the 4th transformation in channel
24410	TRAFO_AXES_IN_4	Axis assignment for the 4th transformation
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Geo-axis assignment for 4th transformation
24426	TRAFO_INCLUDES_TOOL_4	Tool handling with active transformation 4.
24430	TRAFO_TYPE_5	Definition of the 5th transformation in channel
24432	TRAFO_AXES_IN_5	Axis assignment for the 5th transformation

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

6.11.1.3 TRAANG

Channelspecific machine data

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

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

6.11.1.4 Chained transformations

Channelspecific machine data

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

6.11.1.5 Non transformation-specific machine data

Channelspecific machine data

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

6.11.2 Signals

6.11.2.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Transformation active	DB21, ...DBX33.6	DB3300.DBX1.6

M5: Measurement 7

7.1 Brief description

Channel-specific measuring

The trigger event programmed for channel-specific measuring in a part program block initiates the measuring operation and specifies the measurement method used for the measurement. The instructions apply to all axes programmed in this particular block.

Axial measurement

In the case of axial measuring, a measurement can be made from the part program and also from synchronized actions. A measurement method, the encoder and the trigger events are programmed. Whereby the trigger events are formed from the probe (1 and 2) and the trigger criterion (rising/falling signal edge). Depending on the measuring task, several measured values per measurement and trigger event can be recorded.

Preset actual value memory and scratching

The **preset actual value memory** is initiated by means of an HMI operator action. The calculated frame can be written to system frame $P_SETFRAME. The setpoint position of an axis in the WCS can be altered when the actual value memory is preset.

The calculation is performed in the NC when a PI service is activated via

- HMI operator action or a
- Part program command from the measuring cycles.

The term **scratching** refers to both the workpiece measurement **and** the tool measurement. The measurements can be initiated via the

- HMI operator action or via
- Measuring cycles.

Communication with the NC takes place via predefined system variables.

Workpiece and tool measurement

The position of the workpiece can be measured in relation to an edge, a corner or a hole.

To determine the zero position of the workpiece (workpiece zero W) or a hole, setpoint positions can be added to the measured positions in the workpiece coordinate system. The resulting offsets can be entered in a selected frame.

In the case of tool measurement, the control calculates the distance between the tool tip and the tool carrier reference point T from the tool length specified by the user.

Measuring cycles

A description of how to handle measuring cycles can be found in:
Literature:
Programming Manual Measuring cycles

7.2 Hardware requirements

7.2.1 Probes that can be used

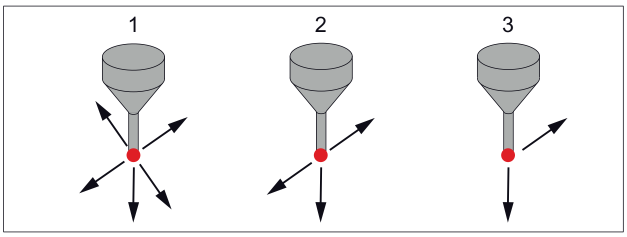
Overview

A switching probe that supplies a constant bounce-free signal on deflection must be used for the "Measuring" function in SINUMERIK controls. Probes that supply only one **pulse** for deflection are **not** suitable.

Probe types

The probe types listed below differ depending on the number of possible deflection directions:

- Multidirectional probe (1)
- Bidirectional probe (2)
- Monodirectional probe (3)



Probe types

Multidirectional probe (3D)

Multidirectional probes can be used unconditionally for measuring tool and workpiece dimensions.

Bidirectional probe

Bidirectional probes are treated like monodirectional probes for the workpiece measurement in milling and machining centers. Bidirectional probes for the workpiece measurement can be used for turning machines.

Monodirectional probe

Monodirectional probes can be used for workpiece measurement on milling machines and machining centers with only slight limitations.

Spindle position and monodirectional probe

The use of monodirectional probes in milling machines and machining centers requires that the spindle can be positioned with the `SPOS` function and the switching signal of the probe transferred over 360°.

The probe must be mechanically aligned in the spindle to permit measurements in the following directions at the 0° spindle position:

Plane	Measurement direction
G17 (X-Y)	Positive X direction
G18 (Z-X)	Positive Z direction
G19 (Y-Z)	Positive Y direction

Note

The complete measurement takes longer because the spindle needs to be positioned several times with `SPOS` during the measuring cycle.

7.3 Channel-specific measuring

7.3.1 Measurement

Activation

The measurement is activated from the part program. A trigger event and a measuring method are programmed.

A distinction is made between two measuring methods:

- MEAS: Measurement with deletion of distance-to-go

  Example:

  N10 G01 F300 X300 Z200 MEAS=-2

  Trigger event is the falling edge (-) of the second probe (2).

- MEAW: Measurement without deletion of distance-to-go

  Example:

  N20 G01 F300 X300 Y100 MEAW=1

  Trigger event is the rising edge of the first probe (1).

The measuring job is aborted with RESET or when the program advances to a new block.

Note

If a geometry axis is programmed in a measuring block, the measured values are stored for all current geometry axes.

If an axis participating in a transformation is programmed in a measurement block, the measured values for all axes participating in this transformation are recorded.

Probe status

It is possible to scan the probe status directly in the part program and in synchronized actions.

$A_PROBE[n] where n= probe

$A_PROBE[n]==1: Probe deflected

$A_PROBE[n]==0: Probe not deflected

7.3.2 Measurement results

Reading measurement results

The results of the measurement commands are stored in system data of the NCK and can be read via system variables in the part program.

- System variable $AC_MEA[No]

  Query measurement job status signal.

  [No] stands for probe (1 or 2)

  The variable is deleted at the beginning of a measurement. The variable is set as soon as the probe fulfills the activation criterion (rising or falling edge). Execution of the measurement job can thus be checked in the part program.

- System variable $AA_MM[axis]

  Access to measured value in the machine coordinate system (MCS)

  Read in part program and in synchronized actions.

  [Axis] stands for the name of the measurement axis (X, Y, ...).

- System variable $AA_MW[axis]

  Access to measured value in the workpiece coordinate system.

  Read in part program and in synchronized actions.

  [Axis] stands for the name of the measurement axis (X, Y, ...).

PLC service display

The functional test for the probe is performed using an NC program.

The measuring signal can be checked at the end of the program in the diagnostic menu "PLC status".

Table 7- 1 Status display for measurement signal

	Status display
Probe 1 deflected	DB10, ...DBX107.0
Probe 2 deflected	DB10, ...DBX107.1

The current measuring status of the axis is displayed by means of the interface signal DB31, ... DBX62.3.

Bit 3=1: Measurement
active Bit 3=0: Measurement not active

This signal can be displayed for all measurement functions and also be read in synchronized actions with

- system variable $AA_MEAACT[axis].

References:
Function Manual, Synchronized Actions

7.4 Axial measurement

7.4.1 Measurement

Activation

Axial measurement can be programmed with and without deletion of distance-to-go. The activation of the measuring is carried out from the part program or a synchronized action. The measuring method and up to four trigger events are programmed. The measuring mode specifies the chronological or programmed sequence of the trigger events. For the continuous measurement, the number of the circular buffer (FIFO) is programmed.

A distinction is made between three measuring methods:

- MEASA: Measurement with deletion of distance-to-go

  Example:

  N10 MEASA[X]=(1,1,-1) G01 X100 F100

  Measuring in mode 1 with active measuring system. Trigger events are the rising and falling edge of the first probe (1) on the travel path to X=100.

  Note

  MEASA cannot be programmed in synchronized actions.

- MEAWA: Measurement without deletion of distance-to-go

  Example:

  N20 MEAWA[X]=(1,-1,1,-2,2) G01 X100 F100

  Measuring in mode 1 with active measuring system. Trigger events are the falling and rising edge of the first probe (1) and of the second probe (2) on the travel path to X=100.

- MEAC (option): Continuous measurement without deletion of distance-to-go

  Example:

  N30 MEAC[X]=(1,1,-2,2) G01 X100 F100

  Measuring in mode 1 with active measuring system. Save the measured values under the first FIFO (1). Trigger events are the falling and rising edge of the second probe (2) on the travel path to X=100.

The measuring job is aborted with RESET or when the program advances to a new block.

Note

MEASA and MEAWA can be programmed in a block. If MEASA/MEAWA are programmed in the same block as MEAS/MEAW, an error message will be output.

Measuring mode

The measuring mode specifies whether trigger events must be activated in parallel or sequentially in ascending sequence and defines the number of measurements to be taken.

- Units decade (measuring mode)

  0 = abort measurement job (e.g. for synchronized actions)

  1 = up to four different trigger events that can be activated at the same time.

  2 = up to four trigger events can be activated in succession.

  Error output if the first trigger event is already present

  3 = up to four trigger events can be activated in succession.

  NO error output if the first trigger event is already present

- Tens decade (measuring system)

  0/not set = use active measuring system

  1 = first measuring system

  2 = second measuring system (if available, otherwise the first measuring system is used. No error message is output).

  3 = first and second measuring system

Note

If the measuring job is performed with two measuring systems, a maximum of two trigger events can be programmed. The measured values of both measuring systems are acquired for each of the two trigger events.

Mode 3 is not supported by MEAC.

Trigger event

A trigger event comprises the number of the probe and the trigger criterion (rising or falling edge) of the measuring signal.

1 = rising edge of probe 1

-1 = falling edge of probe 1

2 = rising edge of probe 2

-2 = falling edge of probe 2

If the measuring operation is performed with two measuring systems, a maximum of two trigger events per probe can be programmed (rising or falling edge). The measured values of both probes are acquired for each of the two trigger events. The default PROFIBUS telegram 391 setting means one measured value per trigger event and position controller cycle is possible.

For MEAC, the number of measured values per trigger event can be increased with PROFIBUS telegram 395 to eight measured values for rising edge and eight measured values for falling edge of each measuring input and position control cycle. This allows higher feed rates or speeds to be implemented compared with PROFIBUS telegram 391.

For further information concerning the telegram selection, see Section "Telegram selection (Page 449)".

FIFO variables

The axial measured values are available in the machine coordinate system (MCS). They are stored in the circular buffer (FIFO) specified by MEAC. The measured values are entered in FIFO variables in the FIFO using the circular principle, e.g. $AC_FIFO1. When two probes have been projected for the measurement, the measured values of the second probe are saved separately in the subsequent FIFO.

The number of measured values is limited. The length of the FIFO can be configured in the following machine data:

MD28264 LEN_AC_FIFO (length of the FIFO variables $AC_FIFO1 - $AC_FIFO10)

The values can be read from the FIFO both in the part program and from synchronized actions.

The measurement remains active until:

- MEAC[axis]=(0) is programmed
- a FIFO is full
- RESET is pressed or end of program M02/M30 is reached

References:
Function Manual, Synchronized Actions; Detailed Description,
Section: Parameters ($AC_FIFO)

Probe status

It is possible to scan the probe status directly in the part program and in synchronized actions.

$A_PROBE[n] where n= probe

$A_PROBE[n]==1: Probe deflected

$A_PROBE[n]==0: Probe not deflected

Probe limitation

The probe limitation is available at the following system variables for PROFIBUS telegram 395:

$A_PROBE_LIMITED[n], where n = probe

$A_PROBE_LIMITED[n]==1: Probe limitation active

$A_PROBE_LIMITED[n]==0: Probe limitation inactive/reset

For additional information on system variables, see:

Reference:
Parameter Manual, System Variables

7.4.2 Telegram selection

Telegram selection for the axial measurement with MEAC

By default, the axial measurement is implemented with PROFIBUS telegram 391. PROFIBUS telegram 395 is used for measuring with several measured values per trigger event and the position control cycle.

The telegram selection is made from the STEP 7 HW Config, in the "DP Slave Properties" > "Configuration" dialog.

The following settings are also required:

- Drive parameters:

  CU: p0922 = 395; setting for the telegram selection

  CU: p0684 = 16; setting of the measuring procedure

  CU: p0680; configuration of the central probe terminal
- PROFIBUS connection:

  MD13211 $MN_MEAS_CENTRAL_SOURCE = 2 (telegram integration without handshake)

7.4.3 Measurement results

Read measurement results of MEASA, MEAWA

The results of the measurement commands are stored in system data of the NCK and can be read via system variables in the part program.

- **System variable $AC_MEA[No]**

  Query measurement job status signal.

  [No] stands for probe (1 or 2)

  The variable is deleted at the beginning of a measurement. The variable is set as soon as the probe fulfills the activation criterion (rising or falling edge). Execution of the measurement job can thus be checked in the part program.
- **System variable $AA_MM1[axis]** to **$AA_MM4[axis]**

  Access to the measured value of the trigger signal in the machine coordinate system. Read in part program and in synchronized actions.

  [Axis] stands for the name of the measurement axis (X, Y, ...).
- **System variable $AA_MW1[axis]** to **$AA_MW4[axis]**

  Access to the measured value of the trigger signal in the workpiece coordinate system. Read in part program and in synchronized actions.

  [Axis] stands for the name of the measurement axis (X, Y, ...).

Two measuring systems

If the measuring job is performed with two measuring systems, a maximum of two trigger events can be programmed. The measured values of both probes are acquired for each of the two probes.

One trigger event

$AA_MM1[axis] = trigger event 1, measured value from encoder 1

$AA_MM2[axis] = trigger event 1, measured value from encoder 2

Two trigger events

$AA_MM1[axis] = trigger event 1, measured value from encoder 1

$AA_MM2[axis] = trigger event 1, measured value from encoder 2

$AA_MM3[axis] = trigger event 2, measured value from encoder 1

$AA_MM4[axis] = trigger event 2, measured value from encoder 2

PLC service display

For further information about the functional test of the probe, see Section "Measurement results (Page 445)".

Read measurement results of MEAC

All measurements are written to a previously defined FIFO variable. The possible number of measured values is defined via the machine data (see Section "Measurement (Page 446)").

- The correct operation is executed reliably for PROFIBUS telegram 391 for the ratios of the IPO clock / position controller cycle ≤ 8:1.
- The correct operation is executed reliably for PROFIBUS telegram 395 for the ratios of the IPO clock / position controller cycle ≤ 4:1.
- The contents of the FIFO memory can be read only once. When measurement results are used more than once, the read-out values must be buffered in the user data.

Endless measuring

In order to implement endless measuring, FIFO values must be read cyclically from the part program. The frequency with which the measured values are read from the FIFO memory and processed must correspond to the writing frequency of the NC.

The number of valid entries is readable in a FIFO variable.

If measuring is to be terminated after reaching a defined number of measured values, the measurement job must explicitly be deselected in the program.

7.5 Setting zeros, workpiece measuring and tool measuring

7.5.1 Preset actual value memory and scratching

Preset actual value memory

Preset actual value memory is initiated by means of an HMI operator action or via measuring cycles. The calculated frame can be written to system frame $P_SETFRAME. The setpoint position of an axis in the WCS can be altered when the actual value memory is preset.

The calculation is performed in the NC when a PI service is activated via

- HMI operator action or a
- Part program command from the measuring cycles.

A tool and a plane can be selected as a basis for the calculation. The calculated frame is entered in the result frame.

Scratching

The term **scratching** refers to both the workpiece **and** tool measurement. The position of the workpiece can be measured in relation to an edge, a corner or a hole. To determine the zero position of the workpiece or the hole, setpoint positions can be added to the measured positions in the WCS. The resulting offsets can be entered in a selected frame. In the tool measurement, the length or radius of a tool can be measured using a measured reference part.

The measurements can be initiated via the

- HMI operator action or via
- Measuring cycles.

Communication with the NC takes place via predefined system variables. The calculation is performed in the NCK when a PI service is activated via:

- the HMI operator action
- or through a part program command from the measuring cycles.

A tool and a plane can be selected as a basis for the calculation. The calculated frame is entered in the result frame.

For more detailed information about channel-specific system frames, please see:

Parameter Manual, System Variables; list of system variables, Section: Frames

Function Manual, Basic Functions; Axes, Coordinate Systems, Frames (K2),
Section: Frames of the frame chain

Additional references:

Programming Manual, Production Planning; Technology Cycles
, Section: Swiveling - CYCLE800

7.5.2 Workpiece measuring

Workpiece measuring

For workpiece measurement, a probe is moved up to the clamped workpiece in the same way as a tool. Due to the variety of different measuring types available, the most common measurement jobs can be performed quite simply and easily on a turning or milling machine.

The position of the workpiece can be measured in relation to an edge, a corner or a hole.

To determine the zero position of the workpiece (workpiece zero W) or a hole, setpoint positions can be added to the measured positions in the WCS. The resulting offsets can be entered in a selected frame.

Variable interface

The variable interface comprises several system variables,

These are categorized as either:

- Input values
- Output values

Reference:
Parameter Manual, System Variables

Input values must be written by the HMI or the cycles. The output values result from the calculations.

References:
Programming Manual, Measuring Cycles

7.5.2.1 Input values

Validity bits for the measurement types

To define which system variables are valid for the current measurement, each measuring process must first declare all the variables as invalid. This is done with:
$AC_MEAS_VALID = 0.

Each input variable implicitly sets the corresponding bit in $AC_MEAS_VALID when writing to the interface. If the validity bits are not reset, the values remain valid for the next calculation.

Note

The interface is not reset in case of machine control table reset or after M30 (reset at program end).

Table 7- 2 Validity bits for the input values of the variables $AC_MEAS_VALID

Bit	Input value	Meaning
0	$AA_MEAS_POINT1[axis]	1. Measuring point for all channel axes
1	$AA_MEAS_POINT2[axis]	2. Measuring point for all channel axes
2	$AA_MEAS_POINT3[axis]	3. Measuring point for all channel axes
3	$AA_MEAS_POINT4[axis]	4. Measuring point for all channel axes
4	$AA_MEAS_SETPOINT[axis]	Setpoint position of the edge, corner, hole
5	$AC_MEAS_WP_SETANGLE	Setpoint workpiece position angle α; -90 < φ < 180
6	$AC_MEAS_CORNER_SETANGLE	Setpoint angle of intersection φ of the corner 0 < φ < 180
7	$AC_MEAS_T_NUMBER	Selected tool
7	$AC_MEAS_D_NUMBER	Selected cutting edge
9	$AC_MEAS_DIR_APPROCH	Approach direction for edge, groove, web and tool measurement only
10	$AC_MEAS_ACT_PLANE	Set working plane and infeed direction
11	$AC_MEAS_FRAME_SELECT	Calculated frame in the specified frame
12	$AC_MEAS_TYPE	Types of workpiece measurement
13	$AC_MEAS_FINE_TRANS	Enter translational offsets
14	$AA_MEAS_SETANGEL[axis]	Setpoint angle of an axis
15	$AA_MEAS_SCALEUNIT	Unit of measurement for input and output values
16	$AA_MEAS_TOOL_MASK	Tool settings
17	$AA_MEAS_P1_COORD	Coordinate system of 1st measuring point
18	$AA_MEAS_P2_COORD	Coordinate system of 2nd measuring point
19	$AA_MEAS_P3_COORD	Coordinate system of 3rd measuring point
20	$AA_MEAS_P4_COORD	Coordinate system of 4th measuring point
21	$AA_MEAS_SET_COORD	Coordinate system of setpoint
22	$AA_MEAS_CHSFR	System frame mask
23	$AA_MEAS_NCBFR	Mask for global basic frame
24	$AA_MEAS_CHBFR	Mask for channel basic frames
25	$AA_MEAS_UIFR	Settable frame from data management
26	$AA_MEAS_PFRAME	Do not calculate programmable frames
27	$AC_MEAS_INPUT[n]	Measuring input parameter with length n

Note

All axis actual values of the appropriate measuring point are invalidated by:

$AC_MEAS_LATCH = 0

Measuring points

A maximum of four measuring points are available for all channel axes for measurement:

Type	Input variable	Meaning
REAL	$AA_MEAS_POINT1[axis]	1. Measuring point for all channel axes
REAL	$AA_MEAS_POINT2[axis]	2. Measuring point for all channel axes
REAL	$AA_MEAS_POINT3[axis]	3. Measuring point for all channel axes
REAL	$AA_MEAS_POINT4[axis]	4. Measuring point for all channel axes

The measured points are normally available as actual values (= setpoint values) in WCS. A measuring point is denoted as valid as soon as an axis value is described for it. Each individual measuring point can be written or picked up.

A few measuring types also support measuring points lying in a different coordinates system (BCS, MCS). The entry in which the coordinates system of the corresponding measuring point was measured can be done via the following variables:

Type	Input variable	Meaning	Values
INT	$AA_MEAS_P1_COORD	Coordinate system of 1st measuring point	0: WCS is the standard setting 1: BCS 2: MCS 3: ENS 4: WCS_REL 5: ENS_REL
INT	$AA_MEAS_P2_COORD	Coordinate system of 2nd measuring point	
INT	$AA_MEAS_P3_COORD	Coordinate system of 3rd measuring point	
INT	$AA_MEAS_P4_COORD	Coordinate system of 4th measuring point	
INT	$AA_MEAS_SET_COORD	Coordinate system of setpoint	

Actual values

The measuring points can be described for all the axes having the current axis actual values. The positions are picked up with reference to the selected coordinates system. The positions are attached in WCS if no coordinates system is specified. The following variable is used for this purpose:
$AC_MEAS_LATCH[0..3]

The index varies from 0 to 3, corresponding to the 1st to 4th measuring point. Assigning the value zero to the variable has the effect that all axis actual values of the corresponding measuring point become invalid. Assigning the value 1 picks up all the axis actual values in the corresponding measuring point. The variable is a write-only variable.

Individual axis actual values of a measuring point can be described with the following variables:

Type	System variable	Meaning	Values
REAL	$AA_MEAS_P1_VALID[ax]	1. Pick up the measuring point of an axis	0: The measuring point of the axis is invalid 1: The measuring point of the axis is determined
REAL	$AA_MEAS_P2_VALID[ax]	2. Pick up the measuring point of an axis	
REAL	$AA_MEAS_P3_VALID[ax]	3. Pick up the measuring point of an axis	
REAL	$AA_MEAS_P4_VALID[ax]	4. Pick up the measuring point of an axis	

The variables $AC_MEAS_LATCH[0..3] and $AA_MEAS_P[1..4]_VALID can be used interactively. Allowance is made accordingly for the facing axis with diameter programming.

Setpoints

The resultant frame is calculated so that the measurement complies with the setpoints specified by the user.

Table 7- 3 Input values for the user setpoint values

Type	System variable	Meaning
REAL	$AA_MEAS_SETPOINT[ax]	Setpoint position of an axis
REAL	$AA_MEAS_SETANGLE[ax]	Setpoint angle of an axis
INT	$AA_MEAS_SP_VALID[ax]	1: Setpoint position of axis is valid / 0: Invalid
REAL	$AC_MEAS_WP_SETANGLE	Rated workpiece position angle α: -90 < α < 180
REAL	$AC_MEAS_CORNER_SETANGLE	Setpoint cutting angle φ of corner: 0 < φ < 180
INT	$AC_MEAS_DIR_APPROACH *)	Approach direction: 0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z

*) The approach direction is required only for the edge, groove, web and tool measurement.

The following measuring points are irrelevant and not evaluated:

- On inputting the setpoint workpiece position angle α: of the 2nd measuring point.
- When inputting the setpoint angle of intersection φ: at the 4th measuring point.

Plane separation

Plane separation for defining the tool orientation. The active level is used for all calculations if no level is specified.

Type	System variable	Values
INT	$AC_MEAS_ACT_PLANE	0: G17 working plane x/y infeed direction z 1: G18 working plane z/x infeed direction y 2: G19 working plane y/z infeed direction x

Translational offsets

When measuring workpieces, translational offsets can be entered in the fine offset component of the selected frame. Variable $AC_MEAS_FINE_TRANS is used for this purpose.

Type	System variable	Values
INT	$AC_MEAS_FINE_TRANS	0: Translational compensation is entered in the coarse offset. 1: Translational compensation is entered in the fine offset.

The following is applicable if the variable $AC_MEAS_FINE_TRANS is not described:

- The compensation value is entered in the coarse offset and transformed in the time frame. There can also be a fine portion in the translation by virtue of the transformations.
- If the following machine data is not preset to 1:

  MD18600 $MN_MM_FRAME_FINE_TRANS

  The compensation is always entered in the course offset.

Calculated frame

When a workpiece is measured, the calculated frame is entered in the specified frame.

Type	System variable	Meaning
INT	$AC_MEAS_FRAME_SELECT	Frame selection during tool measurement

The variable $AC_MEAS_FRAME_SELECT can assume the following values:

Value		Meaning
0	$P_SETFRAME	Active system frame
1	$P_PARTFRAME	Active system frame
2	$P_EXTFRAME	Active system frame
10..25	$P_CHBFRAME[0..15]	Active channel-specific basic frames
50..65	$P_NCBFRAME[0..15]	active NCU-global basic frames
100..199	$P_IFRAME	The calculation is done using the active settable frame, if the corresponding frame is selected. If the selected frame is not active, the corresponding data management frame is included in the calculation.
500	$P_TOOLFRAME	Active system frame
501	$P_WPFRAME	Active system frame
502	$P_TRAFRAME	Active system frame
503	$P_PFRAME	Active current programmable frame
504	$P_CYCFRAME	Active system frame
505	$P_RELFRAME (workpiece coordinate system)	Active system frame
506	$P_RELFRAME (SZS)	Active system frame
1010..1025	$P_CHBFRAME[0..15]	active channel specification Basic frames with active G500
1050..1065	$P_NCBFRAME[0..15]	active NCU-global Basic frames with active G500
2000	$P_SETFR	System frame in data management
2001	$P_PARTFR	System frame in data management
2002	$P_EXTFR	System frame in data management
2010..2025	$P_CHBFR[0..15]	Channel-specific basic frames in data management
2050..2065	$P_NCBFR[0..15]	NCU-global basic frame in data management

Value		Meaning
2100..2199	$P_UIFR[0..99]	settable frame in data management
2500	$P_TOOLFR	System frame in data management
2501	$P_WPFR	System frame in data management
2502	$P_TRAFR	System frame in data management
2504	$P_CYCFR	System frame in data management
2505	$P_RELFR (workpiece coordinate system)	System frame in data management
2506	$P_RELFR (SZS)	System frame in data management
3010..3025	$P_CHBFR[0..15]	Channel-spec. Basic frames with active G500 in data management
3050..3065	$P_NCBFR[0..15]	NCU-global basic frames with active G500 in data management

The MEASURE( ) function calculates frame $AC_MEAS_FRAME according to the specified frame.

In the case of values

0 to 1065, the calculation is performed using the active frame.

2000 to 3065, the calculation is performed with reference to the selected frame in data management. The frame selection in data management is not supported for measurement types 14 and 15. A frame does not have to be active in order to select it in data management. In this case, the calculation is performed as if the frame were active in the chain.

The measuring point is transformed in the selected system and the selected frame is determined using the entire frame including the selected frame. Preset actual value memory is active only after compensation and activation of the frame.

In the case of values

With active **G500** active (1010..1025, 1050..1065, 3010..3025, 3050..3065), the target frame is calculated so that G500 must be active after the frame is selected so that the setpoint position can be calculated.

Conversion into another coordinate system

If a position is to be converted to a position in another coordinate system, the following variables can be used to specify the composition of the desired frame chain:

Type	System variable	Meaning	Values
INT	$AC_MEAS_CHSFR	Selection of system frames	Bit mask corresponding to MD28082 $MC_MM_SYSTEM_FRAME_MASK
INT	$AC_MEAS_NCBFR	Selection of global basic frames	Bit mask (0 ... FFFF)
INT	$AC_MEAS_CHBFR	Selection of channel basic frames	Bit mask (0 ... FFFF)
INT	$AC_MEAS_UIFR	Selection of settable frames	0 ... 99
INT	$AC_MEAS_PFRAME	Programmable frame	0: is included 1: is not included

The data management frames are read and a new frame set up for the corresponding values in the variables.

Note

If variables are not set, the active frames are retained.

Values should only be written to those variables whose data management frames are to be included in the new frame chain. In the case of the basic frames, **only all** of the frames can be exchanged, and not just a particular frame. Active changes via $P_NCBFRMASK and $P_CHBFRMASK are ignored.

Array variable for workpiece and tool measurement

The following array variable of length n is used for further input parameters that are used in the various measurement types

Type	System variable	Meaning	Values
REAL	$AC_MEAS_INPUT[n]	Measurement input parameters	n = 0 ... 9

The control action of the measurement input parameters is described with the measuring methods.

Selection of tool or cutting edge

The tool and edge number of the active tool must correspond to the selected tool. When T0, D0 is selected, the active tool is calculated. If no tool is active, the tool selected by T, D is calculated. No tool other than the selected tool may be active.

Type	System variable	Meaning
INT	$AC_MEAS_T_NUMBER	Selected tool
INT	$AC_MEAS_D_NUMBER	Selected cutting edge

Measurements with 3D probe

When measuring with the 3D probe, the radius of the tool is already compensated with reference to the measuring point, and therefore the radius does not have to be included when calculating the various measurement operations. This property can be defined by means of the following variable:

Type	System variable	Meaning
INT	$AC_MEAS_TOOL_MASK	Tool position

The variable $AC_MEAS_TOOL_SCREEN can assume the following values:

Value	Meaning
0x0	All tool lengths are considered (default setting).
0x1	Tool radius is not included in the calculation
0x2	Tool position in x direction (G19)
0x4	Tool position in y direction (G18)
0x8	Tool position in y direction (G17)
0x10	Tool length is not included in the calculation
0x20	Length of the active tool is included in the coordinate transformation of a position.
0x40	Tool position in x direction.
0x80	Tool position in y direction.
0x100	Tool position in z direction.
0x200	Tool length differential values are included negatively.

Whether or not the radius of a milling tool is included in the calculation can be determined from the tool position and approach direction. If the approach direction is not specified explicitly, it is determined by the selected plane.

Plane	Approach direction
G17	-z direction
G18	- y direction:
G19	- x direction

7.5.2.2 Measurement selection

The measurement is selected by means of the following variable:

Type	System variable	Description
INT	$AC_MEAS_TYPE	Measurement type selection

The variable $AC_MEAS_TYPE can assume the following values:

Value		Description
0		Default
1	Edge_x	Measuring the x edge
2	Edge_y	Measuring the y edge
3	Edge_z	Measuring the z edge
4	Corner_1	Measuring Corner 1
5	Corner_2	Measuring Corner 2
6	Corner_3	Measuring Corner 3
7	Corner_4	Measuring Corner 4
8	Hole	Measuring a hole

Value		Description
9	Stud	Measuring a shaft
10 *	Tool length	Measuring the tool length
11 *	ToolDiameter	Measuring the tool diameter
12	Slot	Measuring a groove
13	Plate	Measuring a web
14	Set_Pos	Preset actual value for geometric and special axes
15	Set_AuxPos	Preset actual value memory for special axes only
16	Edge_2P	Measuring an inclined edge
17	Plane_Angles	Angle of a plane
18	Plane_Normal	Angle of a plane with setpoint input
19	Dimension_1	1-dimensional setpoint value
20	Dimension_2	2-dimensional setpoint value
21	Dimension_3	3-dimensional setpoint value
22 *	ToolMagnifier	ShopTurn: Measurement of tool lengths with zoom-in function
23 *	ToolMarkedPos	ShopTurn: Measuring a tool length with marked position
24	Coordinate transformation	Coordinate transformation of a position
25	Rectangle	Measurement of a rectangle
26	Save	Saving data management frames
27	Restore	Restoring data management frames
28	Taper turning	Additive rotation of the plane

* Types of workpiece measurement

The individual methods are listed under "Types of workpiece measurement" or "Types of tool measurement"and explained in more detail using an appropriate programming example.

7.5.2.3 Output values

Calculation results

If a setpoint position has been specified, the resulting frame is entered in result frame $AC_MEAS_FRAME. This frame can be read and written in the part program. The result frame is calculated according to the selected frame.

If no frame has been selected, the result frame determines the final translation and rotation in the WCS. This frame can be entered in the selected frame using the PI service _N_SETUDT and parameter type no. 7. Once it has been entered, the result frame is deleted.

Table 7- 4 Output values of calculation results

Type	System variable	Description
FRAME	$AC_MEAS_FRAME	Result frame
REAL	$AC_MEAS_WP_ANGLE	Calculated workpiece position angle α

Type	System variable	Description
REAL	$AC_MEAS_CORNER_ANGLE	Calculated angle of intersection ϕ
REAL	$AC_MEAS_DIAMETER	Calculated diameter
REAL	$AC_MEAS_TOOL_LENGTH	Calculated tool length
REAL	$AC_MEAS_RESULTS[10]	Calculation results (depending on $AC_MEAS_TYPE)

7.5.2.4 Calculation method

Activating the calculation

The calculation is activated by an HMI operator action with PI service _N_SETUDT. This PI service can accept one of the following parameter types:

Type	Meaning
1	Active tool offset
2	Active basic frames
3	Active settable frame
4	Global basic frames
5	Global settable frames
6	Calculate workpiece zero or tool lengths
7	Activate workpiece zero (write scratching)
8	Activate external work offset
9	Activate active tool carrier, TCOABS and PAROT

The change becomes apparent immediately in the reset state. In the stop state, the frame is retracted at the next start.

Note

The PI service can be executed only in the reset and stop states. In the case of workpiece measurement, the calculated frame is activated immediately with type no. 7. In the case of tool measurement, the PI service must not be dispatched with type no. 7, since a zero point does not have to be activated.

Activation in the Stop state

The new WCS positions are refreshed in the Stop state. With the continuation start on the part program, the distance-to-go of the interrupted block is deleted. Traversal is made from the current position to the end point of the next block.

Therefore, it is also possible in the Stop state to start a spindle in the MDA mode or in the part program and set and scratch an actual value with M0. Another measurement can also be performed.

Measuring cycles

The calculation in the measuring cycles is performed according to the predefined function:

INT MEASURE( )

MEASURE() delivers a result frame that can be read via $AC_MEAS_FRAME:

- The result is the translation and rotation from the setpoint values recalculated on the selected frame.
- The result frame is calculated as follows:

  The concatenated total frame produces the concatenation of the total frame (prior to measurement) with the calculated translation and rotation.

NOTICE
No preprocessing limitation
MEASURE() does not trigger any implicit block search stop. Because MEASURE( ) works with the frames of the preprocessing block, the user must decide whether a preprocessing stop is required prior to the calculation.

Note

If no frame is selected, the calculated frame is not transformed, i.e. the translation and rotation is determined on the basis of the specified setpoints and the calculated position of the edge, corner, groove, etc. Where the function is used more than once, it is always added to the result frame.

The result frame may need to be deleted beforehand.

Semaphore variable

The measurement variable occurs only once per channel. The measuring operation can be initiated via an operator input in the stop and reset states. The operation can overlap with the measuring cycles in the stop state. The following variable serves to protect against mutual overwriting:

$AC_MEAS_SEMA (semaphore of measurement interface)

The semaphore variable $AC_MEAS_SEMA is

- set to 1 at the beginning of the cycle and
- reset to 0 again at the end of the cycle.

HMI does not use the measurement interface if the variable has the value 1.

Error messages

If the client does not log on, group error number 0xD003 is always generated. If a logon takes place through DIAGN:errCodeSetNrGent or DIAGN:errCodeSetNrPi, then PI_SETUDT provides the error code corresponding to the following syntax:

EX_ERR_PI_REJ_<return value>, e.g. EX_ERR_PI_REJ_ MEASNOTYPE

The following return values are output via the pre-defined MEASURE() function:

Table 7- 5 Predefined error messages

No.	Return values	Meaning
0	MEAS_OK	Correct calculation
1	MEAS_NO_TYPE	Type not specified
2	MEAS_TOOL_ERROR	Error determining the tool
3	MEAS_NO_POINT1	Measuring point 1 does not exist
4	MEAS_NO_POINT2	Measuring point 2 does not exist
5	MEAS_NO_POINT3	Measuring point 3 does not exist
6	MEAS_NO_POINT4	Measuring point 4 does not exist
7	MEAS_NO_SPECPOINT	No reference point available
8	MEAS_NO_DIR	No approach direction
9	MEAS_EQUAL_POINTS	Measuring points are identical
10	MEAS_WRONG_ALPHA	Alpha α is wrong
11	MEAS_WRONG_PHI	Phi ϕ is wrong
12	MEAS_WRONG_DIR	Wrong approach direction
13	MEAS_NO_CROSSING	Lines do not intersect
14	MEAS_NO_PLANE	Planes do not exist
15	MEAS_WRONG_FRAME	No frame or incorrect frame selected
16	MEAS_NO_MEMORY	Insufficient memory available
17	MEAS_INTERNAL_ERROR	Internal error

Tool measurement error

In the case of error code MEAS_TOOL_ERROR or EX_ERR_PI_REJ_MEASTOOLERROR, the system stores a more detailed specification of the error with the following values in output variable $AC_MEAS_TOOL_LENGTH:

Table 7- 6 Predefined error messages for MEAS_TOOL_ERROR

No.	Return values	Meaning
1	TOOL_NO_BLOCK	No block available for the tool calculation
2	TOOL_WRONG_T_NUMBER	Wrong T number
3	TOOL_WRONG_D_NUMBER	Wrong D number
4	TOOL_EVAL_WRONG_TYPE	The tool does not exist
5	TOOL_NO_TOOLCORR_BODY	Memory problem
6	TOOL_DATA_READ_ERROR	Error reading the tool data
7	TOOL_NO_TOOL_WITH_TRAFO	No tool is selected with an active transformation

7.5.2.5 Units of measurement and measurement variables for the calculation

INCH or METRIC unit of measurement

The following input and output variables are evaluated with inch or metric units of measurement:

$AA_MEAS_POINT1[axis]	Input variable for 1st measuring point
$AA_MEAS_POINT2[axis]	Input variable for 2nd measuring point
$AA_MEAS_POINT3[axis]	Input variable for 3rd measuring point
$AA_MEAS_POINT4[axis]	Input variable for 4th measuring point
$AA_MEAS_SETPOINT[axis]	Input variable for setpoint position
$AC_MEAS_DIAMETER	Output variable for calculated diameter
$AC_MEAS_TOOL_LENGTH	Output variable for calculated tool length
$AC_MEAS_RESULTS[n]	Output variable for calculation results

The system of units in which the input and output values can be read or written can be set via the input variable.

INT $AC_MEAS_SCALEUNIT	Unit of measurement for input and output variable
0: Unit of measurement with reference to	active G codes G70/G700 in INCH active G codes G71/G701 in METRIC
1: Unit of measurement corresponds to	the configuration; the measurement system can be set via OPI (standard setting)

The value 1 is always treated as the standard setting if the variable is not written.

Example:

The basic system is metric:

Program code	Comment
G70	
$AC_MEAS_POINT1[x] = $AA_IW[x]	; $AA_IW[x] supplies the basic system
$AC_MEAS_POINT1[x] = 10	; 10 mm
G71	
$AC_MEAS_POINT1[x] = $AA_IW[x]	; $AA_IW[x] supplies the basic system
$AC_MEAS_POINT1[x] = 10	; 10 mm
G700	
$AC_MEAS_POINT1[x] = $AA_IW[x]	; $AA_IW[x] supplies inch value
$AC_MEAS_POINT1[x] = 10	; 10 inch
G710	
$AC_MEAS_POINT1[x] = $AA_IW[x]	; $AA_IW[x] supplies metric value
$AC_MEAS_POINT1[x] = 10	; 10 mm

Diameter programming

Diameter programming is set via machine data:

MD20100 $MC_DIAMETER_AX_DEF = "X"	; Transverse axis is x
MD20150 $MC_GCODE_RESET_VALUES[28] = 2	; DIAMON
MD20360 $MC_TOOL_PARAMETER_DEF_MASK = 'B1001010'	; Tool length, frames and ; Actual value in the diameter

The following is to be taken into account:

- Axis positions in the MCS are not included as diameter value.
- The calculated tool lengths and frame components do not depend on the active G code DIAMON or DIAMOF.
- The measured positions and setpoint positions are read and written depending on DIAMON.
- The translations in the frames are calculated as a diameter in the transverse axis.

Arithmetic and display precision

Position values in mm, inches or degrees are accurately calculated and displayed to six decimal places.

7.5.2.6 Diagnostics

The following diagnostic options exists for the measurement interface:

- If the file /_N_MPF_DIR/_N_MEAS_DUMP_MPF is available, a log is written in the file that should enable the reproduction of the problem.
- The logging is started by creating a blank file having the filename _N_MEAS_DUMP_MPF in the /_N_MPF_DIR directory.
- The content of the file is preserved till it is deleted with $AC_MEAS_VALID = 0.

For runtime reasons, the trace should be activated only if a problem is detected.

7.5.3 Types of workpiece measurement

7.5.3.1 Measurement of an edge (measurement type 1, 2, 3)

Measurement of an x edge ($AC_MEAS_TYPE = 1)

The edge of a clamped workpiece is measured by approaching this edge with a known tool.

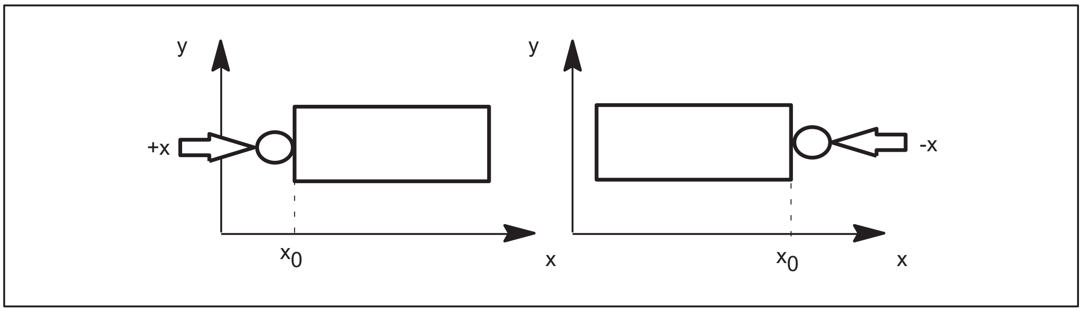


Figure 7-1 x edge

The values of the following variables are evaluated for measurement type 1:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for all channel axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of x edge *
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x
$AC_MEAS_ACT_PLANE	Without specification, calculation is undertaken with the active plane, the radius of the tool is used only in G17 and G18 *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	1

* optional

The following output variables are written for measurement type 1:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of the measured edge

Example

x edge measurement

Program code	Comment
DEF INT RETVAL	
DEF FRAME TMP	
$TC_DP1[1,1]=120	; Type
$TC_DP2[1,1]=20	; 0
$TC_DP3[1,1]=10	; (z) length compensation vector
$TC_DP4[1,1]= 0	; (y)
$TC_DP5[1,1]=0	; (x)
$TC_DP6[1,1]=2	; Radius
T1 D1	
g0 x0 y0 z0 f10000	
G54	
	; Measure x edge
$AC_MEAS_VALID = 0	; Set all input values to invalid
g1 x-1 y-3	; 1. Approach measuring point
$AA_MEAS_POINT1[x] = $AA_IW[x]	
$AA_MEAS_POINT1[y] = $AA_IW[y]	
$AA_MEAS_POINT1[z] = $AA_IW[z]	
$AC_MEAS_DIR_APPROACH = 0	; Set approach direction +x
$AA_MEAS_SETPOINT[x] = 0	; Set setpoint position of the edge
$AA_MEAS_SETPOINT[y] = 0	
$AA_MEAS_SETPOINT[z] = 0	
$AC_MEAS_ACT_PLANE = 0	; Measuring plane is G17
$AC_MEAS_FRAME_SELECT = 101	; Select frame - IFRAME
$AC_MEAS_T_NUMBER = 1	; Select tool
$AC_MEAS_D_NUMBER = 1	
$AC_MEAS_TYPE = 1	; Set measurement type for x edge
RETVAL = MEASURE()	; Start measuring process

Program code	Comment
if RETVAL <> 0	
setal(61000 + RETVAL)	
endif	
$P_IFRAME = $AC_MEAS_FRAME	
$P_UIFR[1] = $P_IFRAME	; Describe system frame in data management
g1 x0 y0	**; Approach the edge**
m30	

Measurement of a y edge ($AC_MEAS_TYPE = 2)

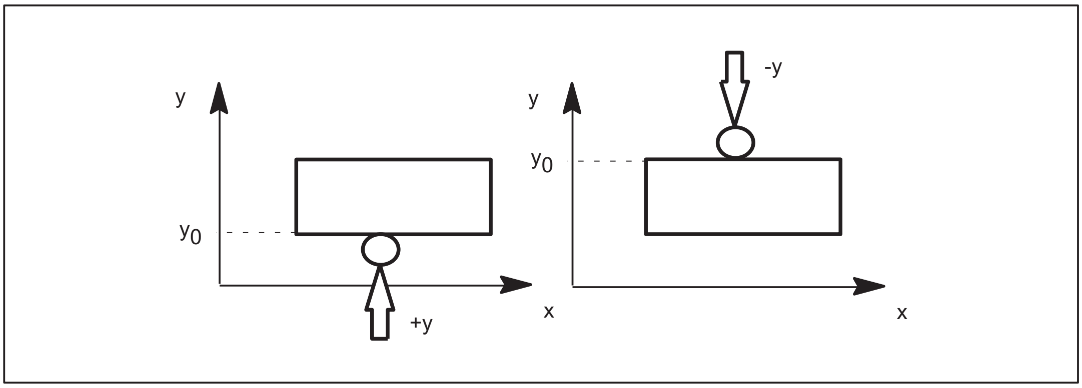


Figure 7-2 y edge

The values of the following variables are evaluated for measurement type 2:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for all channel axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of y edge *
$AC_MEAS_DIR_APPROACH	2: +y, 3: -y
$AC_MEAS_ACT_PLANE	Without specification, calculation is undertaken with the active plane, the radius of the tool is used only in G17 and G19 *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	2

* optional

The following output variables are written for measurement type 2:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of the measured edge

Measurement of a z edge ($AC_MEAS_TYPE = 3)

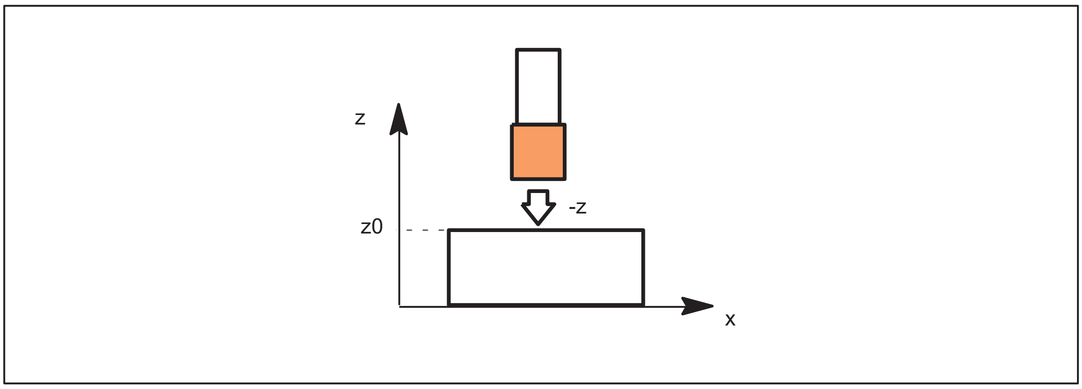


Figure 7-3 z edge

The values of the following variables are evaluated for measurement type 3:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for all channel axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of z edge *
$AC_MEAS_DIR_APPROACH	4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Without specification, calculation is undertaken with the active plane, the radius of the tool is used only in G18 and G19 *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	3

* optional

The following output variables are written for measurement type 3:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of the measured edge

7.5.3.2 Measurement of an angle (measurement type 4, 5, 6, 7)

Measurement of a corner C1 - C4 ($AC_MEAS_TYPE = 4, 5, 6, 7)

A corner is uniquely defined by approaching four measuring points P1 to P4. Three measuring points suffice for a known angle of intersection ϕ.

If the angle of intersection ϕ and the workpiece position angle α are known, two measurement points P1 and P3 suffice.

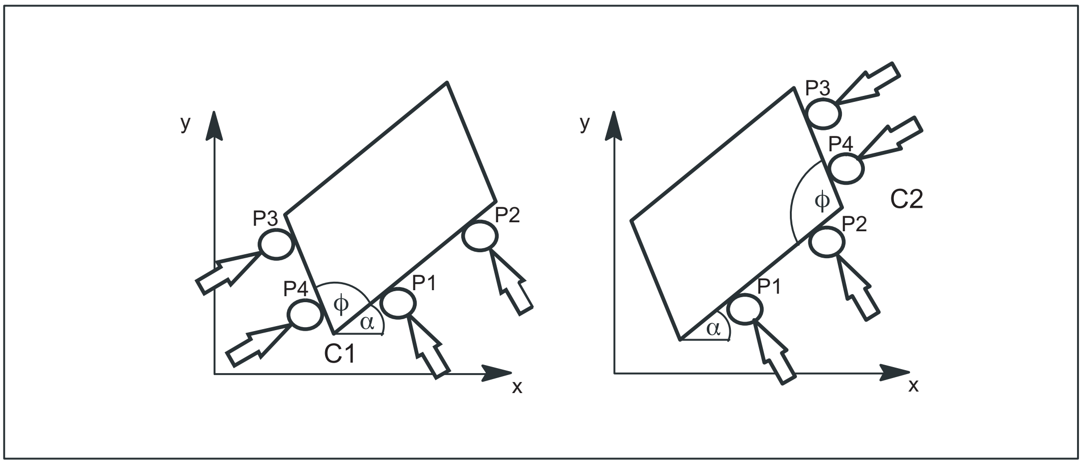


Figure 7-4 Corner C1, corner C2

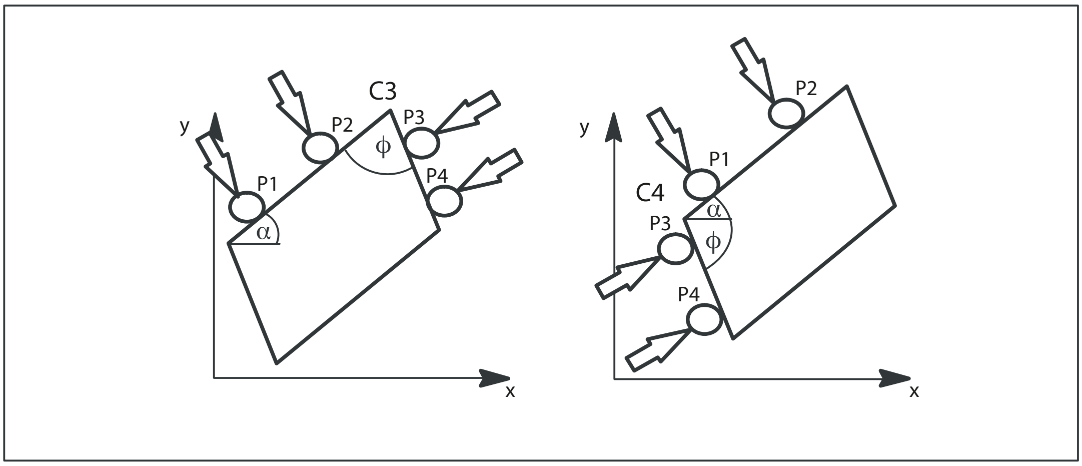


Figure 7-5 Corner C3, corner C4

The values of the following variables are evaluated for measurement types 4 to 7:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2 irrelevant for $AC_MEAS_WP_SETANGLE

Input variable	Meaning
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_POINT4[axis]	Measuring point 4 irrelevant for $AC_MEAS_CORNER_SETANGLE
$AA_MEAS_WP_SETANGLE	Setpoint workpiece position angle *
$AA_MEAS_CORNER_SETANGLE	Setpoint angle of intersection *
$AA_MEAS_SETPOINT[axis]	Setpoint position of corner *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Without specification of outer corner * =0: Measurement for outer corner =1: Measurement for inner corner
$AC_MEAS_TYPE	4, 5, 6, 7

* optional

The following variables are written for measurement types 4 to 7:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation and rotation
$AC_MEAS_WP_ANGLE	Calculated workpiece position angle
$AC_MEAS_CORNER_ANGLE	Calculated angle of intersection
$AC_MEAS_RESULTS[0]	Abscissa of calculated vertex
$AC_MEAS_RESULTS[1]	Ordinate of calculated vertex
$AC_MEAS_RESULTS[2]	Applicate of calculated vertex

Example

Corner measurement C1: Corner with three measuring points (P1, P3 and P4) and known angle of intersection ϕ (90°) and unknown workpiece position angle α.

```
Program code                              Comment
DEF INT RETVAL
DEF FRAME TMP

$TC_DP1[1,1]=120                          ; Type
$TC_DP2[1,1]=20                           ; 0
$TC_DP3[1,1]=10                           ; (z) length compensation vector
$TC_DP4[1,1]=0                            ; (y)
```

Program code	Comment
$TC_DP5[1,1]=0	; (x)
$TC_DP6[1,1]=2	; Radius
T1 D1	
g0 x0 y0 z0 f10000	
G54	
$P_CHBFRAME[0] = crot(z,45)	
$P_IFRAME[x,tr] = -sin(45)	
$P_IFRAME[y,tr] = -sin(45)	
$P_PFRAME[z,tr] = -45	
	; **Measure corner with 3 measuring points**
$AC_MEAS_VALID = 0	; Set all input values to invalid
g1 x-1 y-3	; 1. Approach measuring point
$AC_MEAS_LATCH[0] = 1	; Pick up measuring point P1
g1 x-4 y4	; 3. Approach measuring point
$AC_MEAS_LATCH[2] = 1	; Pick up measuring point P3
g1 x-4 y1	; 4. Approach measuring point
$AC_MEAS_LATCH[3] = 1	; Pick up measuring point P4
$AA_MEAS_SETPOINT[x] = 0	; Set position setpoint of the corner to (0, 0, 0)
$AA_MEAS_SETPOINT[y] = 0	
$AA_MEAS_SETPOINT[z] = 0	
$AC_MEAS_CORNER_SETANGLE = 90	; Define setpoint angle of intersection φ
$AC_MEAS_ACT_PLANE = 0	; Measuring plane is G17
$AC_MEAS_FRAME_SELECT = 0	; Select frame - SETFRAME
$AC_MEAS_T_NUMBER = 1	; Select tool
$AC_MEAS_D_NUMBER = 1	
$AC_MEAS_TYPE = 4	; Set measuring type on corner 1
RETVAL = MEASURE()	; Start measuring process
if RETVAL <> 0	
setal(61000 + RETVAL)	
endif	

Program code	Comment
`if $AC_MEAS_CORNER_ANGLE <> 90`	`; Query known setpoint angle of intersection φ`
`setal(61000 + $AC_MEAS_CORNER_ANGLE)`	
`endif`	
`$P_SETFRAME = $AC_MEAS_FRAME`	
`$P_SETFR = $P_SETFRAME`	`; Describe system frame in data management`
`g1 x0 y0`	**`; Approach the corner`**
`g1 x10`	`; Approach the rectangle`
`y10`	
`x0`	
`y0`	
`m30`	

7.5.3.3 Measurement of a hole (measurement type 8)

Measuring points for determining a hole ($AC_MEAS_TYPE = 8)

Three measuring points are needed to determine the center point and diameter. The three points must all be different. When four points are specified, the circle is adjusted in accordance with the least square method. The circle is determined so that the sum of the distance squares of the points to the circle is minimal. The quality of the adjustment can be read.

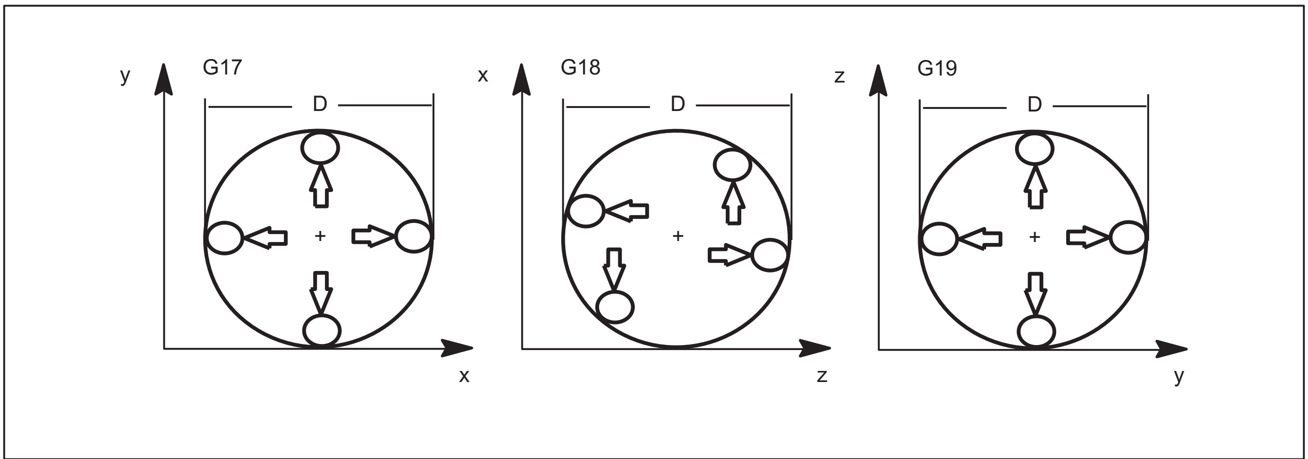


Figure 7-6 Hole

The values of the following variables are evaluated for measurement type 8:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_POINT4[axis]	When specified, the center is determined from four points *
$AA_MEAS_SETPOINT[axis]	Setpoint position of hole center *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	8

* optional

The following output variables are written for measurement type 8:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_DIAMETER	Diameter of hole
$AC_MEAS_RESULTS[0]	Abscissa of the calculated center point
$AC_MEAS_RESULTS[1]	Ordinate of the calculated center point
$AC_MEAS_RESULTS[2]	Applicate of the calculated center point
$AC_MEAS_RESULTS[3]	Quality of the circle adjustment: Sum of the distance squares

Example

Measuring a hole

```
Program code                          Comment
DEF INT RETVAL
DEF FRAME TMP

$TC_DP1[1,1]=120                      ; Type
$TC_DP2[1,1]=20                       ; 0
$TC_DP3[1,1]=10                       ; (z) length compensation vector
$TC_DP4[1,1]=0                        ; (y)
$TC_DP5[1,1]=0                        ; (x)
$TC_DP6[1,1]=2                        ; Radius
```

Program code	Comment
T1 D1	
g0 x0 y0 z0 f10000	
G54	
	: **Measure hole**
$AC_MEAS_VALID = 0	; Set all input values to invalid
g1 x-3 y0	; 1. Approach measuring point
$AA_MEAS_POINT1[x] = $AA_IW[x]	
$AA_MEAS_POINT1[y] = $AA_IW[y]	
$AA_MEAS_POINT1[z] = $AA_IW[z]	
g1 x0 y3	; 2. Approach measuring point
$AA_MEAS_POINT2[x] = $AA_IW[x]	
$AA_MEAS_POINT2[y] = $AA_IW[y]	
$AA_MEAS_POINT2[z] = $AA_IW[z]	
g1 x3 y0	; 3. Approach measuring point
$AA_MEAS_POINT3[x] = $AA_IW[x]	
$AA_MEAS_POINT3[y] = $AA_IW[y]	
$AA_MEAS_POINT3[z] = $AA_IW[z]	
$AA_MEAS_SETPOINT[x] = 0	; Set setpoint position of the center
$AA_MEAS_SETPOINT[y] = 0	
$AA_MEAS_SETPOINT[z] = 0	
$AC_MEAS_ACT_PLANE = 0	; Measuring plane is G17
$AC_MEAS_FRAME_SELECT = 0	; Select frame - SETFRAME
$AC_MEAS_T_NUMBER = 1	; Select tool
$AC_MEAS_D_NUMBER = 1	
$AC_MEAS_TYPE = 8	; **Set measuring type on hole**
RETVAL = MEASURE()	; Start measuring process
if RETVAL <> 0	
setal(61000 + RETVAL)	
endif	
if $AC_MEAS_DIAMETER <> 10	; Query known diameter
setal(61000 + $AC_MEAS_WP_ANGLE)	
endif	

Program code	Comment
`$P_SETFRAME = $AC_MEAS_FRAME`	
`$P_SETFR = $P_SETFRAME`	`; Describe system frame in data management`
`g1 x-3 y0`	`; **Approach P1**`
`g2 I = $AC_MEAS_DIAMETER / 2`	`; Approach hole in reference to the center of the circle`
`m30`	

7.5.3.4 Measurement of a shaft (measurement type 9)

Measuring points for determining a shaft ($AC_MEAS_TYPE = 9)

Three measuring points are needed to determine the center point and diameter. The three points must all be different. When four points are specified, the circle is adjusted in accordance with the least square method. The circle is determined so that the sum of the distance squares of the points to the circle is minimal. The quality of the adjustment can be read.

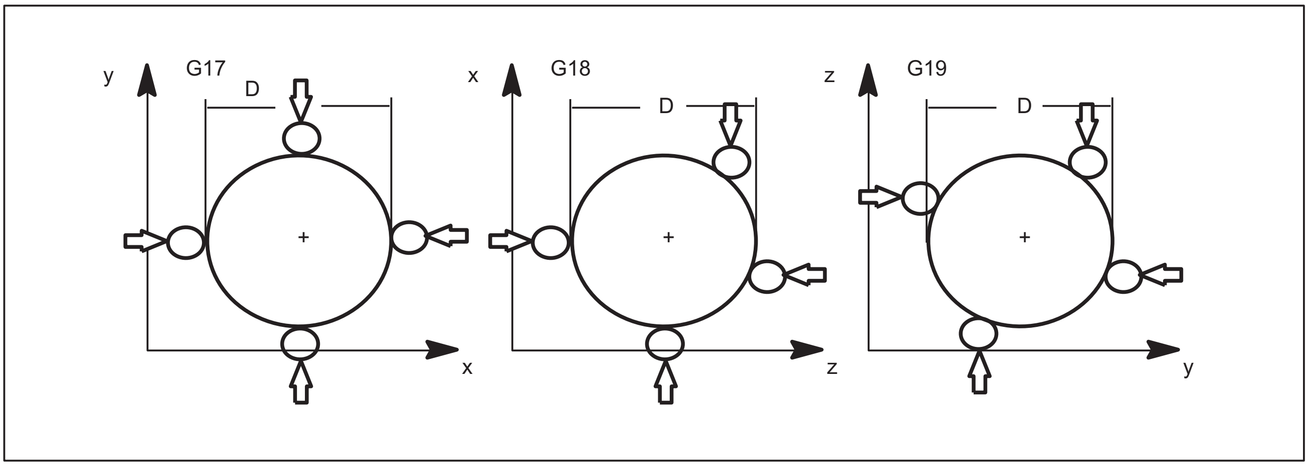


Figure 7-7 Shaft

The values of the following variables are evaluated for measurement type 9:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3

Input variable	Meaning
$AA_MEAS_POINT4[axis]	When specified, the center is determined from four points *
$AA_MEAS_SETPOINT[axis]	Setpoint position of shaft center point *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	9

* optional

The following output variables are written for measurement type 9:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_DIAMETER	Diameter of hole
$AC_MEAS_RESULTS[0]	Abscissa of the calculated center point
$AC_MEAS_RESULTS[1]	Ordinate of the calculated center point
$AC_MEAS_RESULTS[2]	Applicate of the calculated center point
$AC_MEAS_RESULTS[3]	Quality of the circle adjustment: Sum of the distance squares

7.5.3.5 Measurement of a groove (measurement type 12)

Measuring points for determining the position of a groove ($AC_MEAS_TYPE = 12)

A groove is measured by approaching the two outside corners or inner edges. The groove center can be set to a setpoint position. The component of the approach direction determines the groove position.

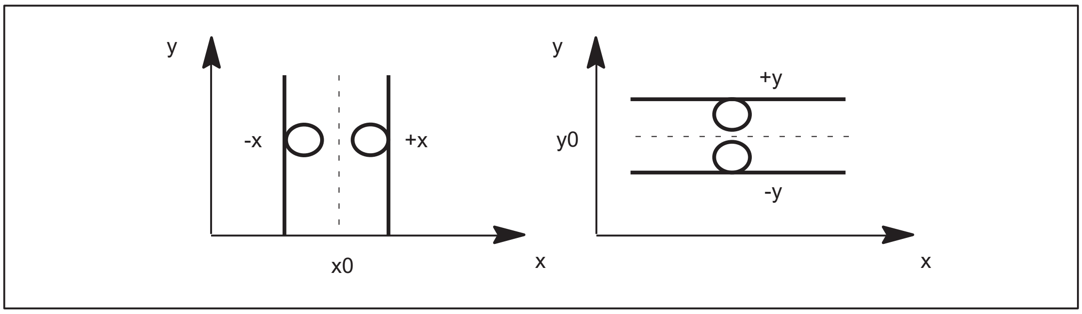


Figure 7-8 Groove

The values of the following variables are evaluated for measurement type 12:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_SETPOINT[axis]	Setpoint position of groove center *
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Approach direction for 2nd measuring point for a recess measurement. Must have the same coordinate as the approach direction of the 1st point. * 0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_TYPE	12

* optional

The following output variables are written for measurement type 12:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of calculated groove center (x0, y0 or z0)
$AC_MEAS_RESULTS[1]	Groove width in approach direction

Example

Groove measurement with approach direction in x

```
Program code                         Comment
DEF INT RETVAL
DEF FRAME TMP

$TC_DP1[1,1]=120                     ; Type
$TC_DP2[1,1]=20                      ; 0
$TC_DP3[1,1]=10                      ; (z) length compensation vector
$TC_DP4[1,1]=0                       ; (y)
$TC_DP5[1,1]=0                       ; (x)
$TC_DP6[1,1]=2                       ; Radius

T1 D1
g0 x0 y0 z0 f10000
G54
```

Program code	Comment
$P_CHBFRAME[0] = crot(z,45)	
$P_IFRAME[x,tr] = -sin(45)	
$P_IFRAME[y,tr] = -sin(45)	
$P_PFRAME[z,rt] = -45	
	; Measure groove
$AC_MEAS_VALID = 0	; Set all input values to invalid
g1 x-2	; 1. Approach measuring point
$AA_MEAS_POINT1[x] = $AA_IW[x]	
$AA_MEAS_POINT1[y] = $AA_IW[y]	
$AA_MEAS_POINT1[z] = $AA_IW[z]	
g1 x4	; 2. Approach measuring point
$AA_MEAS_POINT2[x] = $AA_IW[x]	
$AA_MEAS_POINT2[y] = $AA_IW[y]	
$AA_MEAS_POINT2[z] = $AA_IW[z]	
$AA_MEAS_SETPOINT[x] = 0	; Set setpoint position of the groove center
$AA_MEAS_SETPOINT[y] = 0	
$AA_MEAS_SETPOINT[z] = 0	
$AC_MEAS_DIR_APPROACH = 0	; Set approach direction +x
$AC_MEAS_ACT_PLANE = 0	; Measuring plane is G17
$AC_MEAS_FRAME_SELECT = 0	; Select frame - SETFRAME
$AC_MEAS_T_NUMBER = 1	; Select tool
$AC_MEAS_D_NUMBER = 1	
$AC_MEAS_TYPE = 12	**; Set measuring type on groove**
RETVAL = MEASURE()	; Start measuring process
if RETVAL <> 0 setal(61000 + RETVAL)	
endif	
$P_SETFRAME = $AC_MEAS_FRAME	
$P_SETFR = $P_SETFRAME	; Describe system frame in data management
g1 x0 y0	**; Approach the groove center**
m30	

7.5.3.6 Measurement of a web (measurement type 13)

Measuring points for determining the position of a web ($AC_MEAS_TYPE = 13)

A web is measured by approaching the two outside corners or inner edges. The web center can be set to a setpoint position. The component of the approach direction determines the web position.

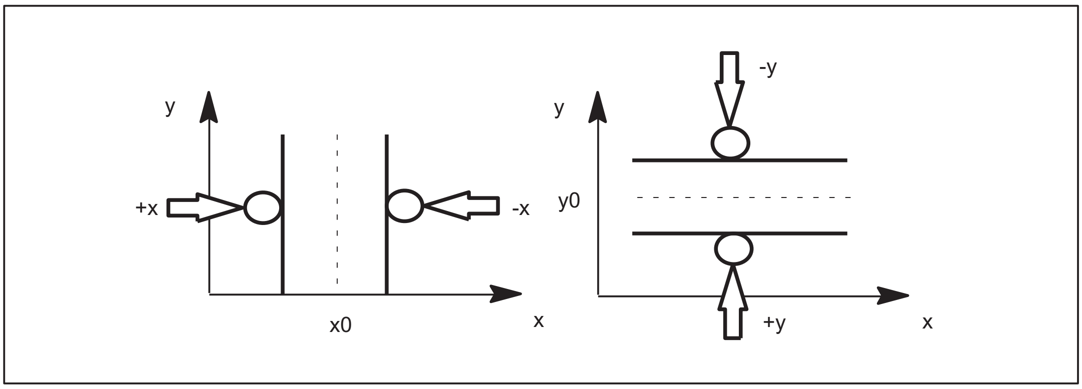


Figure 7-9 Web

The values of the following variables are evaluated for measurement type 13:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_SETPOINT[axis]	Setpoint position of web center *
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Approach direction for 2nd measuring point for a recess measurement. Must have the same coordinate as the approach direction of the 1st point. * 0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_TYPE	13

* optional

The following output variables are written for measurement type 13:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of calculated web center (x0, y0 or z0)
$AC_MEAS_RESULTS[1]	Web width in approach direction

7.5.3.7 Measurement of geo axes and special axes (measurement type 14, 15)

Preset actual value memory for geo axes and special axes ($AC MEAS TYPE = 14)

This measurement type is used on the HMI operator interface.

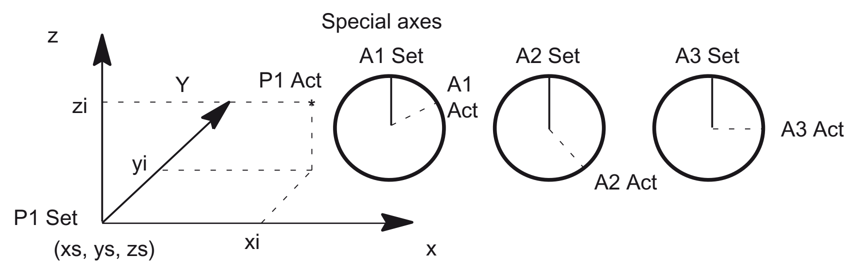


Figure 7-10 Preset actual value memory

The values of the following variables are evaluated for measurement type 14:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Actual values of the axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of the individual axes *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	14

* optional

The following output variables are written for measurement type 14:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translations

Example

Reference point setting in relative coordinate systems.

Program code	Comment
DEF INT RETVAL	
T1 D1	; Activate probe
G54	; Activate all frames and G54
TRANS x=10	; Offset between WCS and ENS
G0 x0 f10000	; WCS(x) = 0; ENS(x) = 10
$AC_MEAS_VALID = 0	; Set all input variables to invalid
$AC_MEAS_TYPE = 14	; **Measuring type for preset actual value memory**
$AC_MEAS_ACT_PLANE = 0	; Measuring plane is G17
$AC_MEAS_P1_COORD = 5	; ENS_REL for 1st measuring point
$AC_MEAS_LATCH[0] = 1	; Pick up all axis positions
$AC_MEAS_SET_COORD = 5	; Setpoint position is relative to ENS
$AA_MEAS_SETPOINT[x] = 0	; Setpoint position in the relative ENS coordinate system
$AC_MEAS_FRAME_SELECT = 2505	; $P_RELFR
RETVAL = MEASURE()	; Calculation of $P_RELFR; PI SETUDT(6)
IF RETVAL <> 0 GOTOF ERROR	
ENDIF $ P_RELFR = $AC_MEAS_FRAME	; Activation; PI SETUDT(7)

Preset actual value memory for special axes only ($AC MEAS TYPE = 15)

This measurement type is used on the HMI operator interface.

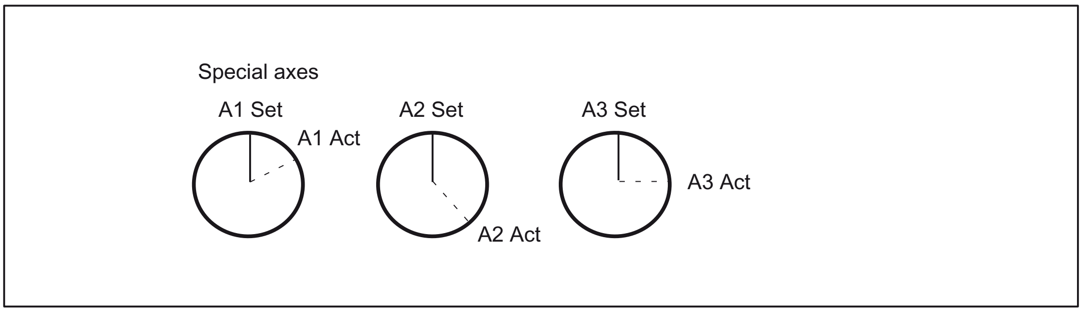


Figure 7-11 Preset actual value memory for special axes only

The values of the following variables are evaluated for measurement type 15:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Actual values of the axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of the individual axes *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_TYPE	15

* optional

The following output variables are written for measurement type 15:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translations

7.5.3.8 Measurement of an oblique edge (measurement type 16)

Measurement of an oblique edge ($AC_MEAS_TYPE = 16)

This measurement determines the position angle of the workpiece and enters it in the frame. A setpoint angle in the +/- 90 degrees range can be input. This can be interpreted as the resultant rotation of the workpiece after the result frame for the active WCS has been activated.

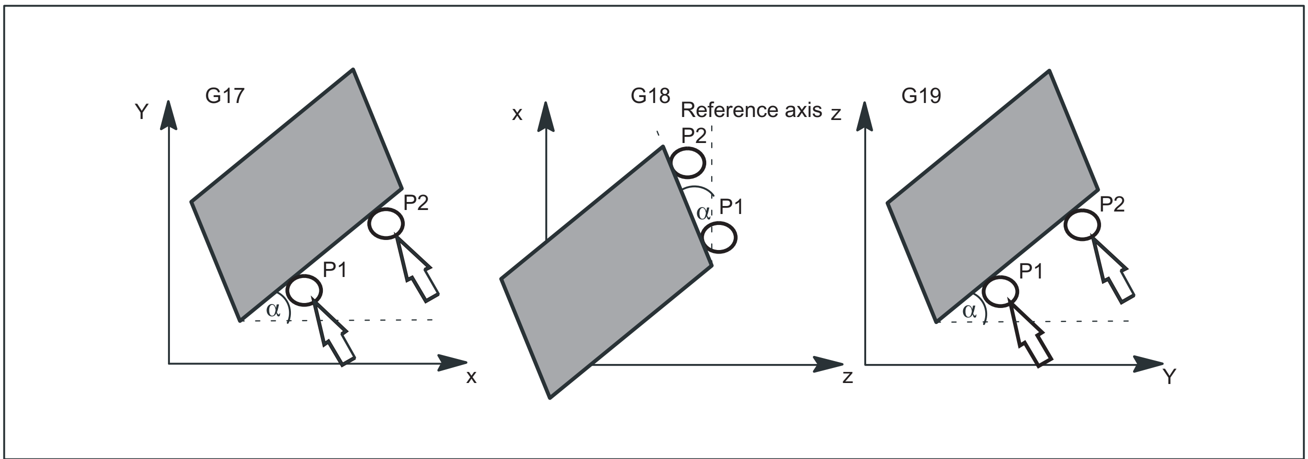


Figure 7-12 Oblique edge in planes G17, G18 and G19

The values of the following variables are evaluated for measurement type 16:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_SETANGLE	Setpoint angle *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Unless otherwise specified, the reference coordinate for the alignment of the workpiece is always the abscissa of the selected plane. * =0: Reference coordinate is the abscissa =1: Reference coordinate is the ordinate
$AC_MEAS_INPUT[1]	Unless otherwise specified, the workpiece position angle is entered in the frame as a rotation. Otherwise, a channel axis index can be specified for a rotary axis whose translation is set to the current rotary axis position plus the calculated rotation. The workpiece is then aligned at rotary axis position = 0. The current rotary axis value must be set in $AA_MEAS_POINT[axis]. *
$AC_MEAS_TYPE	16

* optional

The following output variables are written for measurement type 16:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with rotation
$AC_MEAS_WP_ANGLE	Calculated workpiece position angle

7.5.3.9 Measurement of an oblique angle in a plane (measurement type 17)

Measurement of an angle in an inclined plane ($AC_MEAS_TYPE = 17)

The oblique plane is determined using three measuring points P1, P2 and P3.

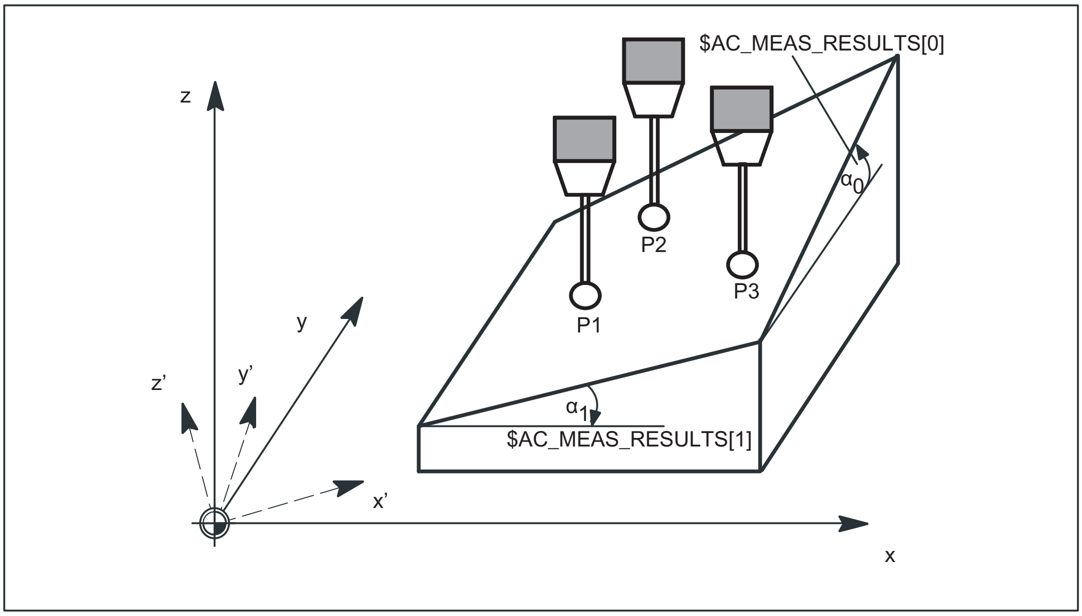


Figure 7-13 Oblique plane in G17

$AC_MEAS_TYPE = 17 defines two resulting angles α_0 and α_1 for the skew of the plane; these are entered in $AC_MEAS_RESULTS[0..1]:

- $AC_MEAS_RESULTS[0] → Rotation at the abscissa
- $AC_MEAS_RESULTS[1] → Rotation at the ordinate

These angles are calculated by means of the three measuring points P1, P2 and P3. In this type of measurement the angle for the applicate ($AC_MEAS_RESULTS[2]) is always pre-filled with 0.

A setpoint rotation that is entered in the result frame can be input for the abscissa and/or the ordinate. If only one angle is specified with a setpoint, the second angle is calculated such that the three measuring points are on an oblique plane with the setpoint angle. Only rotations are entered in the result frame, the WCS reference point is retained. The WCS is rotated such that z' is perpendicular to the oblique plane.

The following plane settings are defined for measurement type 17:

Axis identifier	G17	G18	G19
Abscissa	x axis	z axis	y axis
Ordinate	y axis	x axis	z axis
Applicate (infeed axis)	z axis	y axis	x axis

The values of the following variables are evaluated for measurement type 17:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_SETANGLE[axis]	Setpoint rotations around abscissa and ordinate *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Unless otherwise specified, the points are not projected in a plane * 0: Points are not projected on a plane 1: Points are projected in the active plane or in the selected plane
$AC_MEAS_TYPE	17

* optional

The following output variables are written for measurement type 17:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame
$AC_MEAS_RESULTS[0]	Angles around abscissa from which three measuring points are calculated
$AC_MEAS_RESULTS[1]	Angles around ordinate from which three measuring points are calculated
$AC_MEAS_RESULTS[2]	Angles around applicate from which three measuring points are calculated
$AC_MEAS_RESULTS[3]	Angle around abscissa which is entered in the result frame
$AC_MEAS_RESULTS[4]	Angle around ordinate which is entered in the result frame
$AC_MEAS_RESULTS[5]	Angle around applicate which is entered in the result frame

Example

Measure angle of a plane.

```
Program code                            Comment
DEF INT RETVAL
DEF AXIS _XX, _YY, _ZZ

T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54

$AC_MEAS_VALID = 0                      ; Set all input values to invalid
```

Program code	Comment
$AC_MEAS_TYPE = 17	**; Set measurement type for oblique plane**
$AC_MEAS_ACT_PLANE = 0	; Measuring plane is G17
_XX=$P_AXN1	; Define axes according to the plane
_YY=$P_AXN2	
_ZZ=$P_AXN3	
G17 G1 _XX=10 _YY=10 F1000	; 1. Approach measuring point
MEAS = 1 _ZZ=...	
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]	; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]	; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]	; Assign measurement value to applicate
G1 _XX=20 _YY=10 F1000	; 2. Approach measuring point
MEAS = 1 _ZZ=...	
$AA_MEAS_POINT2[_xx] = $AA_MW[_xx]	; Assign measurement value to abscissa
$AA_MEAS_POINT2[_yy] = $AA_MW[_yy]	; Assign measurement value to ordinate
$AA_MEAS_POINT2[_zz] = $AA_MW[_zz]	; Assign measurement value to applicate
G1 _XX=20 _YY=20 F1000	; 3. Approach measuring point
MEAS = 1 _ZZ=...	
$AA_MEAS_POINT3[_xx] = $AA_MW[_xx]	; Assign measurement value to abscissa
$AA_MEAS_POINT3[_yy] = $AA_MW[_yy]	; Assign measurement value to ordinate
$AA_MEAS_POINT3[_zz] = $AA_MW[_zz]	; Assign measurement value to applicate
	; Define setpoints for angle
$AA_MEAS_SETANGLE[_xx] = 12	; Rotation around the abscissa
$AA_MEAS_SETANGLE[_yy] = 4	; Rotation around the ordinate
$AC_MEAS_FRAME_SELECT = 102	; Select target frame - G55
$AC_MEAS_T_NUMBER = 1	; Select tool
$AC_MEAS_D_NUMBER = 1	
RETVAL = MEASURE()	; Start measurement calculation
if RETVAL <> 0	
setal(61000 + RETVAL)	
endif	
if $AC_MEAS_RESULTS[0] <> 12	
setal(61000 + $AC_MEAS_RESULTS[0])	
endif	

Program code	Comment
if $AC_MEAS_RESULTS[1] <> 4	
setal(61000 + $AC_MEAS_RESULTS[1])	
endif	
$P_UIFR[2] = $AC_MEAS_FRAME	; Write measurement frame in data management (G55)
G55 G0 AX[_xx]=10 AX[_yy]=10	; Activate frame and traverse
m30	

7.5.3.10 Redefine measurement around a WCS reference frame (measurement type 18)

Redefine WCS coordinate system ($AC_MEAS_TYPE = 18)

The zero point of the new WCS is determined by measuring point P1 at surface normal on the oblique plane.

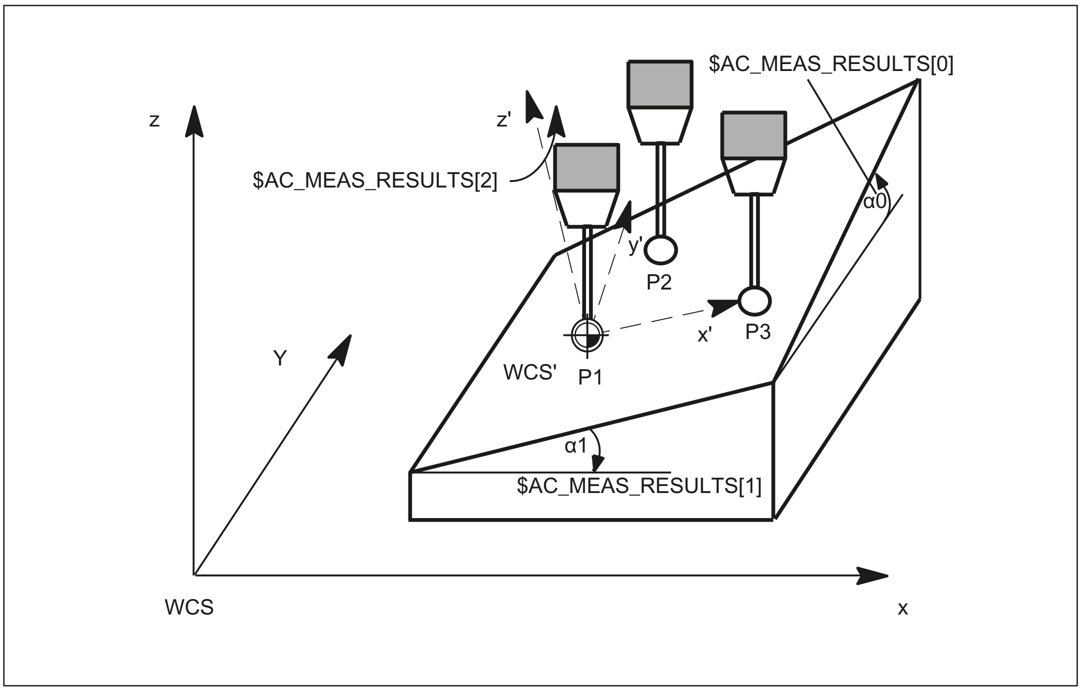


Figure 7-14 Oblique plane in G17

Measurement of plane

The plane is measured in one measuring cycle. The cycle records the three measuring points and passes the necessary values to the variable interface.

The MEASURE() function calculates the solid angles and translational offset of the new WCS' on the basis of the input values.

Transformation of measuring frame

The results of the calculation, i.e. the solid angles and translation, are entered in measuring frame $AC_MEAS_FRAME. The measuring frame is transformed according to the selected frame in the frame chain. The solid angles are also stored in the output values $AC_MEAS_RESULTS[0..2]. In

- The angle around the abscissa of the old WCS is stored in $AC_MEAS_RESULTS[0],
- The angle around the ordinate is stored in $AC_MEAS_RESULTS[1] and
- The angle around the applicate is stored in $AC_MEAS_RESULTS[2].

Define the new WCS' zero

After performing the calculation, the measuring cycle can write and activate the selected frame in the frame chain with the measuring frame. After activation, the new WCS is positioned at surface normal on the inclined plane, with measuring point P1 as the zero point of the new WCS.

The programmed positions then refer to the inclined plane.

Application

CAD systems often define inclined planes by specifying three points P1, P2 and P3 on this plane. In this case,

- 1. measuring point P1 is applied as the new WCS' reference point,
- 2. Measuring point P2 defines the direction of the abscissa x' of the newly rotated WCS' coordinate system and the
- 3. measuring point P3 is used to determine the solid angles.

The values of the following variables are evaluated for measurement type 18:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_SETPOINT[axis]	Setpoint position of P1 *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *

Input variable	Meaning
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Unless otherwise specified, the points are not projected in a plane * 0: Points are not projected on a plane 1: Points are projected in the active plane or in the selected plane
$AC_MEAS_TYPE	18

* optional

The following output variables are written for measurement type 18:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with rotations and transformation
$AC_MEAS_RESULTS[0]	Calculated angle around the abscissa
$AC_MEAS_RESULTS[1]	Calculated angle around the ordinate
$AC_MEAS_RESULTS[2]	Calculated angle around the applicate

Example

Workpiece coordinate system on the inclined plane

```
Program code                              Comment
DEF INT RETVAL
DEF AXIS _XX, _YY, _ZZ

T1 D1                                     ; Activate probe
G54                                       ; Activate all frames and G54

$AC_MEAS_VALID = 0                        ; Set all input values to invalid
$AC_MEAS_TYPE = 18                        ; Set measurement type for oblique plane
$AC_MEAS_ACT_PLANE = 0                    ; Measuring plane is G17

_XX=$P_AXN1                               ; Define axes according to the plane
_YY=$P_AXN2
_ZZ=$P_AXN3

G17 G1 _XX=10 _YY=10 F1000                ; 1. Approach measuring point
MEAS = 1 _ZZ=...

$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]        ; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]        ; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]        ; Assign measurement value to applicate
```

Program code	Comment
G1 _XX=20 _YY=10 F1000	; 2. Approach measuring point
MEAS = 1 _ZZ=...	
$AA_MEAS_POINT2[_xx] = $AA_MW[_xx]	; Assign measurement value to abscissa
$AA_MEAS_POINT2[_yy] = $AA_MW[_yy]	; Assign measurement value to ordinate
$AA_MEAS_POINT2[_zz] = $AA_MW[_zz]	; Assign measurement value to applicate
G1 _XX=20 _YY=20 F1000	; 3. Approach measuring point
MEAS = 1 _ZZ=...	
$AA_MEAS_POINT3[_xx] = $AA_MW[_xx]	; Assign measurement value to abscissa
$AA_MEAS_POINT3[_yy] = $AA_MW[_yy]	; Assign measurement value to ordinate
$AA_MEAS_POINT3[_zz] = $AA_MW[_zz]	; Assign measurement value to applicate
$AA_MEAS_SETPOINT[_xx] = 10	; Define setpoints for P1
$AA_MEAS_SETPOINT[_yy] = 10	
$AA_MEAS_SETPOINT[_zz] = 10	
$AC_MEAS_FRAME_SELECT = 102	; Select target frame - G55
$AC_MEAS_T_NUMBER = 1	; Select tool
$AC_MEAS_D_NUMBER = 1	
RETVAL = MEASURE()	; Start measurement calculation
if RETVAL <> 0	
setal(61000 + RETVAL)	
endif	
	; Calculation results for the solid angles
	; Angle around the …
R0 = $AC_MEAS_RESULTS[0]	; Abscissa for the old WCS
R1 = $AC_MEAS_RESULTS[1]	; Ordinate
R2 = $AC_MEAS_RESULTS[2]	; Applicate
$P_UIFR[2] = $AC_MEAS_FRAME	; Write measurement frame in data management (G55)
G55 G0 AX[_xx]=10 AX[_yy]=10	; Activate frame and traverse
m30	

7.5.3.11 Measurement of a 1-, 2- and 3-dimensional setpoint selection (measurement type 19, 20, 21)

1-dimensional setpoint value ($AC_MEAS_TYPE = 19)

With this measurement method, it is possible to define exactly one setpoint for the abscissa, the ordinate and the applicate. If two or three setpoints are defined, only the first is accepted in the sequence abscissa, ordinate and applicate. The tool is not taken into account.

It is purely an actual value memory preset for the abscissa, the ordinate or the applicate.

The values of the following variables are evaluated for measurement type 19:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for the abscissa
$AA_MEAS_POINT1[axis]	Measuring point 1 for the ordinate
$AA_MEAS_POINT1[axis]	Measuring point 1 for the applicate
$AA_MEAS_SETPOINT[axis]	Setpoint position of abscissa or ordinate or applicate
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_FINE_TRANS	Unless otherwise specified, frame is written to coarse translation *
$AC_MEAS_TYPE	19

* optional

The following output variables are written for measurement type 19:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with rotations and translation

Example

1-dimensional setpoint selection

```
Program code                            Comment
DEF INT RETVAL
DEF REAL _CORMW_XX,
_CORMW_YY,
_CORMW_ZZ
DEF AXIS _XX, _YY, _ZZ
T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54
$AC_MEAS_VALID = 0                      ; Set all input values to invalid
$AC_MEAS_TYPE = 19                      ; Set measurement type for 1-dimensional
                                          setpoint selection
```

Program code	Comment
`$AC_MEAS_ACT_PLANE = 0`	`; Measuring plane is G17`
`_XX=$P_AXN1`	`; Define axes according to the plane`
`_YY=$P_AXN2`	
`_ZZ=$P_AXN3`	
	`; Assign measured values`
`$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]`	`; Assign measurement value to abscissa`
`$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]`	`; Assign measurement value to ordinate`
`$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]`	`; Assign measurement value to applicate`
`$AA_MEAS_SETPOINT[_xx] = 10`	`; Define setpoint for abscissa`
`$AC_MEAS_FRAME_SELECT = 102`	`; Select target frame - G55`
`RETVAL = MEASURE()`	`; Start measurement calculation`
`if RETVAL <> 0`	
`setal(61000 + RETVAL)`	
`endif`	
`$P_UIFR[2] = $AC_MEAS_FRAME`	`; Write measurement frame in data management (G55)`
`G55 G0 AX[_xx]=10 AX[_yy]=10`	`; Activate frame and traverse`
`m30`	

2-dimensional setpoint value ($AC_MEAS_TYPE = 20)

Setpoints for two dimensions can be defined using this measuring method. Any combination of 2 out of 3 axes is permissible. If three setpoints are specified, only the values for the abscissa and the ordinate are accepted. The tool is not taken into account.

This is a purely actual value memory preset.

The values of the following variables are evaluated for measurement type 20:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for the abscissa
$AA_MEAS_POINT1[axis]	Measuring point 1 for the ordinate
$AA_MEAS_POINT1[axis]	Measuring point 1 for the applicate
$AA_MEAS_SETPOINT[axis]	Setpoint position for the 1st dimension
$AA_MEAS_SETPOINT[axis]	Setpoint position for the 2nd dimension
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_FINE_TRANS	Unless otherwise specified, frame is written to coarse translation *
$AC_MEAS_TYPE	20

* optional

The following output variables are written for measurement type 20:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with rotations and translation

Example

2-dimensional setpoint selection

Program code	Comment
DEF INT RETVAL	
DEF REAL _CORMW_XX,	
_CORMW_YY,	
_CORMW_ZZ	
DEF AXIS _XX, _YY, _ZZ	
T1 D1	; Activate probe
G54	; Activate all frames and G54
$AC_MEAS_VALID = 0	; Set all input values to invalid
$AC_MEAS_TYPE = 20	; **Set measurement type for 2-dimensional setpoint selection**
$AC_MEAS_ACT_PLANE = 0	; Measuring plane is G17
_XX=$P_AXN1	; Define axes according to the plane
_YY=$P_AXN2	
_ZZ=$P_AXN3	
	; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]	; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]	; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]	; Assign measurement value to applicate
$AA_MEAS_SETPOINT[_xx] = 10	; Define setpoint for abscissa and ordinate
$AA_MEAS_SETPOINT[_yy] = 10	
$AC_MEAS_FRAME_SELECT = 102	; Select target frame - G55
RETVAL = MEASURE()	; Start measurement calculation
if RETVAL <> 0	
setal(61000 + RETVAL)	
endif	
$P_UIFR[2] = $AC_MEAS_FRAME	; Write measurement frame in data management (G55)
G55 G0 AX[_xx]=10 AX[_yy]=10	; Activate frame and traverse
m30	

3-dimensional setpoint value ($AC_MEAS_TYPE = 21)

Using this measurement method, it is possible to define a setpoint for the abscissa, the ordinate and the applicate. The tool is not taken into account.

It is purely an actual value memory preset for the abscissa, ordinate and applicate.

The values of the following variables are evaluated for measurement type 21:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for the abscissa
$AA_MEAS_POINT1[axis]	Measuring point 1 for the ordinate
$AA_MEAS_POINT1[axis]	Measuring point 1 for the applicate
$AA_MEAS_SETPOINT[axis]	Setpoint position for the abscissa
$AA_MEAS_SETPOINT[axis]	Setpoint position for the ordinate
$AA_MEAS_SETPOINT[axis]	Setpoint position for the applicate
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_FINE_TRANS	Unless otherwise specified, frame is written to coarse translation *
$AC_MEAS_TYPE	21

* optional

The following output variables are written for measurement type 21:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with rotations and translation

Example

3-dimensional setpoint selection

Program code	Comment
`DEF INT RETVAL`	
`DEF REAL _CORMW_XX,`	
`_CORMW_YY,`	
`_CORMW_ZZ`	
`DEF AXIS _XX, _YY, _ZZ`	
`T1 D1`	`; Activate probe`
`G54`	`; Activate all frames and G54`
`$AC_MEAS_VALID = 0`	`; Set all input values to invalid`
`$AC_MEAS_TYPE = 21`	**`; Set measurement type for 3-dimensional setpoint selection`**
`$AC_MEAS_ACT_PLANE = 0`	`; Measuring plane is G17`
`_XX=$P_AXN1`	`; Define axes according to the plane`

Program code	Comment
_YY=$P_AXN2	
_ZZ=$P_AXN3	
	; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]	; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]	; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]	; Assign measurement value to applicate
$AA_MEAS_SETPOINT[_xx] = 10	; Define setpoint for abscissa, ordinate and applicate
$AA_MEAS_SETPOINT[_yy] = 10	; Define
$AA_MEAS_SETPOINT[_zz] = 10	
$AC_MEAS_FRAME_SELECT = 102	; Select target frame - G55
$AA_MEAS_SETPOINT[_yy] = 10	
RETVAL = MEASURE()	; Start measurement calculation
if RETVAL <> 0	
setal(61000 + RETVAL)	
endif	
$P_UIFR[2] = $AC_MEAS_FRAME	; Write measurement frame in data management (G55)
G55 G0 AX[_xx]=10 AX[_yy]=10	; Activate frame and traverse
m30	

7.5.3.12 Measuring a measuring point in any coordinate system (measurement type 24)

Measurement method for converting a measuring point in any coordinate system ($AC_MEAS_TYPE = 24)

With this method of measurement, a measuring point in any coordinate system (WCS, BCS, MCS) can be converted with reference to a new coordinate system by coordinate transformation.

The new coordinate system is generated by specifying a desired frame chain.

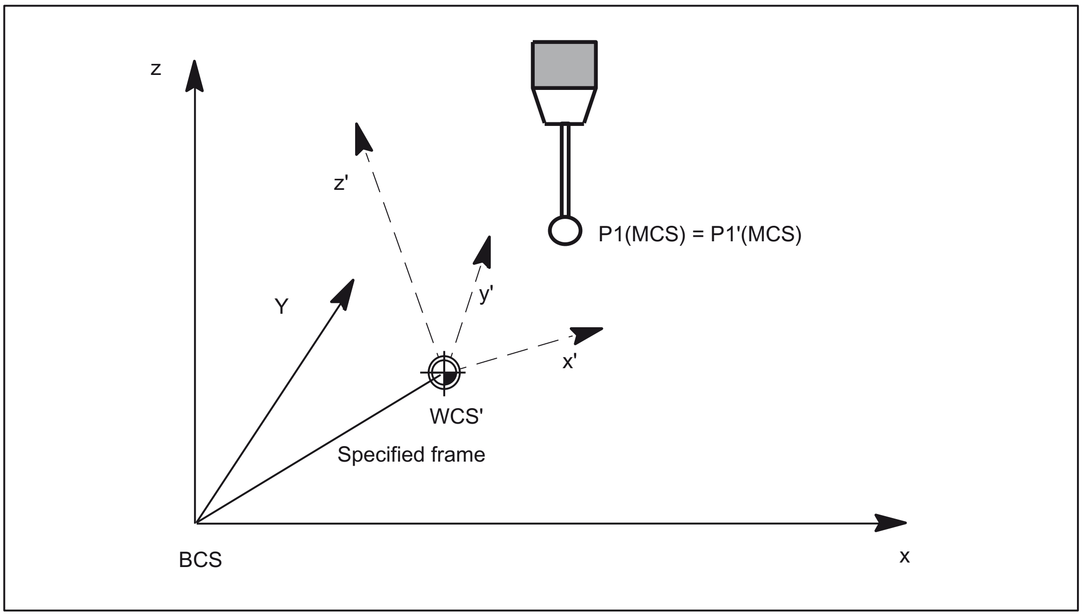


Figure 7-15 Coordinate transformation of a position

The values of the following variables are evaluated for measurement type 24:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Position to be transformed
$AC_MEAS_P1:COORD	Default is 0: WCS, 1: BCS, 2: MCS *
$AC_MEAS_P2_COORD	Target coordinate system *
$AC_MEAS_TOOL_MASK	0x20; Length of the active tool is included in the coordinate transformation of a position *
$AC_MEAS_CHSFR	System frames from data management *
$AC_MEAS_NCBFR	Global basic frames from the data management *
$AC_MEAS_CHBFR	Channel basic frames from the data management *
$AC_MEAS_UIFR	Settable frame from data management *
$AC_MEAS_PFRAME	1: Programmable frame is not included in calculation *
$AC_MEAS_TYPE	24

* optional

The following output variables are written for measurement type 24:

Output variable	Meaning
$AC_MEAS_POINT2[axis]	Converted axis positions

Example

WCS coordinate transformation of a measured position

Program code	Comment
DEF INT RETVAL	
DEF INT LAUF	
DEF REAL_CORMW_xx, _CORMW_yy, _CORMW_zz	
DEF AXIS _XX, _YY, _ZZ	
$TC_DP1[1,1]=120	; Tool type end mill
$TC_DP2[1,1]=20	
$TC_DP3[1,1]=0	; (z) length compensation vector
$TC_DP4[1,1]=0	; (y) length compensation vector
$TC_DP5[1,1]=0	; (x) length compensation vector
$TC_DP6[1,1]=2	; Radius
T1 D1	; Activate probe
G17	; Oblique plane G17
_xx=$P_AXN1 _yy=$P_AXN2 _zz=$P_AXN3	; **Define axes according to the plane**
	; Entire frame results in CTRANS(_xx,10,_yy,-1,_zz,5,A,6,B,7)
$P_CHBFR[0]=CTRANS(_zz,5,A,6) : CROT(_zz,45)	
$P_UIFR[1]=CTRANS( )	
$P_UIFR[1,_xx,TR]=-SIN(45)	
$P_UIFR[1,_yy,TR]=-SIN(45)	
$P_UIFR[2]=CTRANS( )	
$P_PFRAME=CROT(_zz,-45)	
$P_CYCFR=CTRANS(_xx,10,B,7)	
G54	; Activate all frames and G54
G0 X0 Y0 Z0 A0 B0 F1000	
$AC_MEAS_VALID = 0	; Set all input values to invalid
$AC_MEAS_TYPE = 24	; **Set measurement type for coordinate transformation**
$AC_MEAS_ACT_PLANE = 0	; Measuring plane is G17
	; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_IW[_xx]	; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_IW[_yy]	; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_IW[_zz]	; Assign measurement value to applicate
$AA_MEAS_POINT1[A] = $AA_IW[A]	
$AA_MEAS_POINT1[B] = $AA_IW[B]	

Program code	Comment
`$AC_MEAS_P1_COORD=0`	`; Converting a position from WCS into WCS'`
`$AC_MEAS_P2_COORD=0`	
	`;` **`Set WCS`**
	`; Entire frame results in CTRANS(_xx,0,_yy,0,_zz,5,A,6,B,0)`
	`; Stop cycle frame`
`$AC_MEAS_CHSER=$MC_MM_SYSTEM_FRAME_MASK B_AND 'B1011111'`	
`$AC_MEAS__NCBFR='B0'`	`; Stop global basic frame`
`$AC_MEAS__CHBFR='B1'`	`; Channel basic frame 1 from data management`
`$AC_MEAS__UIFR=2`	`; Settable frame G55 from data management`
`$AA_MEAS_PFRAME=1`	`; Do not include programmable frame in calculation`
`RETVAL = MEASURE()`	`; Start measurement calculation`
`if RETVAL <> 0`	
`setal(61000 + RETVAL)`	
`endif`	
`if $AA_MEAS_PIONT2[_xx] <> 10`	
`setal(61000)`	
`M0`	
`stopre`	
`endif`	
`if $AA_MEAS_PIONT2[_yy] <> -1`	
`setal(61000)`	
`M0`	
`stopre`	
`if $AA_MEAS_PIONT2[_zz] <> 0`	
`setal(61000)`	
`M0`	
`stopre`	
`if $AA_MEAS_PIONT2[A] <> 0`	
`setal(61000)`	
`M0`	
`stopre`	
`if $AA_MEAS_PIONT2[B] <> 7`	
`setal(61000)`	
`M0`	
`stopre`	
`m30`	

7.5.3.13 Measurement of a rectangle (measurement type 25)

Measuring points for determining a rectangle ($AC_MEAS_TYPE = 25)

To determine a rectangle, tool dimensions are required in the following working planes.

- G17 working plane x/y infeed direction z
- G18 working plane z/x infeed direction y
- G19 working plane y/z infeed direction x

Four measuring points are required per rectangle.

Measuring points can be specified in any desired order. The measuring points with the largest ordinate distance correspond to points P3 and P4.

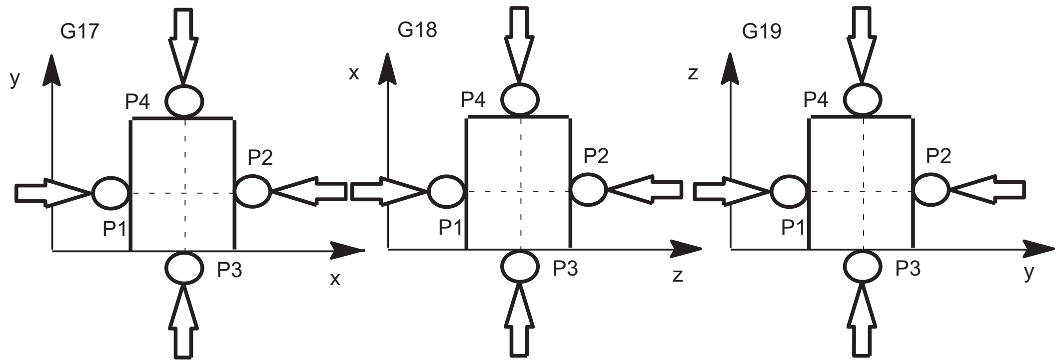


Figure 7-16 Determining a rectangle with infeed into the working plane G17, G18 and G19

The values of the following variables are evaluated for measurement type 25:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_POINT4[axis]	Measuring point 4
$AA_MEAS_SETPOINT[axis]	Setpoint position of web center *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *

Input variable	Meaning
$AC_MEAS_INPUT[0]	Without specification of outer corner * =0: Measurement for outer corner =1: Measurement for inner corner
$AC_MEAS_TYPE	25

* optional

The following output variables are written for measurement type 25:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Abscissa of the calculated center point
$AC_MEAS_RESULTS[1]	Ordinate of the calculated center point
$AC_MEAS_RESULTS[2]	Applicate of the calculated center point
$AC_MEAS_RESULTS[3]	Width of rectangle P1/P2
$AC_MEAS_RESULTS[4]	Length of rectangle P3/P4

7.5.3.14 Measurement for saving data management frames (measurement type 26)

Saving data management frames ($AC_MEAS_TYPE = 26)

This measurement type offers the option of saving some or all data management frames with their current value assignments to a file. The measurement can be initiated by executing a command or via the part program. The function can also be activated from different channels. The files are set up in directory _N_SYF_DIR.

A Restore operation deletes the backed-up data and a new Save operation overwrites the existing back-up. The data last saved can then be deleted with

- $AC_MEAS_CHSFR = 0 system frames;
- $AC_MEAS_NCBFR = 0 global basic frames;
- $AC_MEAS_CHBFR = 0 channel basic frames;
- $AC_MEAS_UIFR = 0 number of settable frames

from the data management system using a second Save operation.

Note

If you decide to create a backup of all data management frames, remember that 1 KB of memory is needed to save each frame. If insufficient memory is available, the process is aborted with error message MEAS_NO_MEMORY. The following machine data item can be used to change the size of the static memory:

MD18351 $MM_DRAM_FILE_MEM_SIZE

The values of the following variables are evaluated for measurement type 26:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AC_MEAS_CHSFR	Bit mask system frames from data management. * If this variable is not written, all system frames are backed up.
$AC_MEAS_NCBFR	Bit mask of global basic frames from the data management. * If this variable is not written, all global basic frames are backed up.
$AC_MEAS_CHBFR	Bit mask of channel basic frames from the data management. * If this variable is not written, all channel basic frames are backed up.
$AC_MEAS_UIFR	Number of settable frames from data management. * 0..100: 1: G500 2: G500, G54. If this variable is not written, all settable frames are backed up.
$AC_MEAS_TYPE	26

* optional

7.5.3.15 Measurement for restoring backed-up data management frames (measurement type 27)

Restoring data management frames last backed up ($AC_MEAS_TYPE = 27)

This measurement type allows data management frames backed up by measurement type 26 to be restored to the SRAM.

It is possible to restore either some or all of the frames last backed up. If a frame that has not been backed up is selected, the selection is ignored. The process is not aborted.

The values of the following variables are evaluated for measurement type 27:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AC_MEAS_CHSFR	Bit mask system frames from data management. * If this variable is not written, all system frames are restored.
$AC_MEAS_NCBFR	Bit mask of global basic frames from the data management. * If this variable is not written, all global basic frames are restored.
$AC_MEAS_CHBFR	Bit mask of channel basic frames from the data management. * If this variable is not written, all channel basic frames are restored.

Input variable	Meaning
$AC_MEAS_UIFR	Number of settable frames from data management. * Range of 1: G54 to G99: G599. If this variable is not written, all settable frames are restored.
$AC_MEAS_TYPE	27

* optional

7.5.3.16 Measurement for defining an additive rotation for taper turning (measurement type 28)

Additive rotation of the plane for taper turning ($AC_MEAS_TYPE = 28)

Measuring type 28 can be used to specify an additive rotation through an angle in the range of α = +/- 90° of the active or a certain plane. The rotation takes place on the coordinate axis at right angles to the plane.

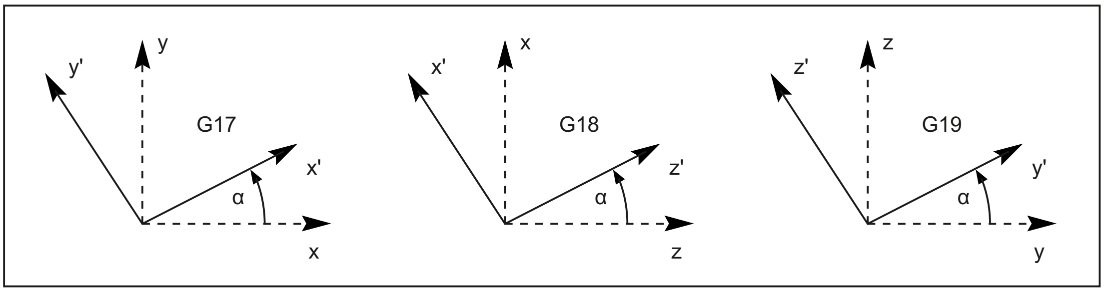


Figure 7-17 Rotation of the planes G17, G18 and G19 by angle α = +30°

Application

With taper turning, the active plane is rotated by the taper angle, whereby the rotation is written in the active cycle frame. With RESET, the cycle frame is deleted. Re-activation may be necessary. The selection of the cycle frame is made depending on the SZS position display. If after activation of the rotation, e.g. with active plane G18, traversing is performed in the direction of z', the actual positions of the corresponding axes change **simultaneously** for **x and z**

Rotations with active planes G17 and G19 behave similarly and are displayed in the above figure. The values of the following variables are evaluated for measurement type 28:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AC_MEAS_WP_SETANGLE	Setpoint angle
$AC_MEAS_ACT_PLANE	Rotation is through the active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_INPUT[0]	1: Taper turning is active. *
$AC_MEAS_TYPE	28

* optional

The following output variables are written for measurement type 28:

Output variable	Meaning
$AC_MEAS_FRAME	Result with rotation

7.5.4 Tool measuring

The control calculates the distance between the tool tip and the tool carrier reference point T from the tool length specified by the user.

The following measurement types can be used to measure a tool loaded on a turning or milling machine:

Measurement types	Tool measuring
$AC_MEAS_TYPE = 10	Tool lengths on a reference part that has already been measured
$AC_MEAS_TYPE = 11	Tool diameter on a reference part that has already been measured
$AC_MEAS_TYPE = 22	Tool diameters on machines with zoom-in function (ShopTurn)
$AC_MEAS_TYPE = 23	Tool lengths with stored or current positions (ShopTurn) Measurement of a tool length of two tools with the following orientation: Two turning tools with: Their own reference point each for tool orientation in the approach direction. One reference point for tool position that is opposite to the approach direction and tool orientation. Two milling tools with: Their own reference point each for tool orientation in -y. One reference point for tool position in -y and a tool position opposite to the approach direction. Two milling tools rotated 90 degrees with: Their own reference point each for tool orientation in the approach direction. One reference point for a tool position that is opposite to the approach direction and tool orientation.

7.5.5 Types of workpiece measurement

7.5.5.1 Measurement of tool lengths (measurement type 10)

Tool length measurement on a reference part that has already been measured ($AC_MEAS_TYPE = 10)

The tool length can be measured on a previously measured reference part.

The plane selection depends on the position of the tool:

- G17 for tool position in the z direction
- G18 for tool position in the y direction
- G19 for tool position in the x direction

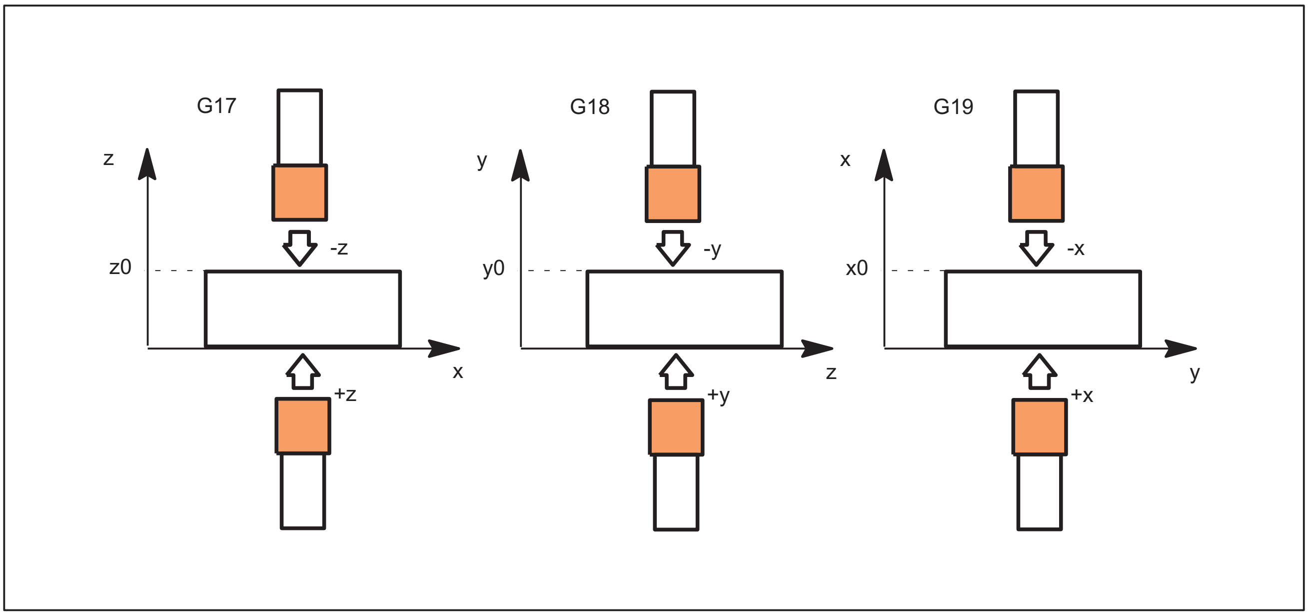


Figure 7-18 Tool length measurement for the selected plane G17, G18 and G19

The values of the following variables are evaluated for measurement type 10:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AC_MEAS_P1_COORD	Coordinate system of the measuring point *
$AA_MEAS_SETPOINT[axis]	Set position z0
$AC_MEAS_SET_COORD	Coordinate system of setpoint *
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z

Input variable	Meaning
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_TYPE	10

* optional

The following output variables are written for measurement type 10:

Output variable	Meaning
$AC_MEAS_TOOL_LENGTH	Tool length
$AC_MEAS_RESULTS[0]	Tool length in x
$AC_MEAS_RESULTS[1]	Tool length in y
$AC_MEAS_RESULTS[2]	Tool length in z
$AC_MEAS_RESULTS[3]	Tool length L1
$AC_MEAS_RESULTS[4]	Tool length L2
$AC_MEAS_RESULTS[5]	Tool length L3

Example

Measuring the tool length

```
Program code                              Comment
DEF INT RETVAL
T0 D0
g0 x0 y0 z0 f10000
                                          ; Measure tool length
$AC_MEAS_VALID = 0                        ; Set all input values to invalid
g1 z10                                    ; Move tool towards reference part

$AC_MEAS_LATCH[0] = 1                     ; Pick up measuring point 1
$AC_MEAS_DIR_APPROACH = 5                 ; Set approach direction -z
$AA_MEAS_SETPOINT[x] = 0                  ; Set reference position
$AA_MEAS_SETPOINT[y] = 0
$AA_MEAS_SETPOINT[z] = 0
$AC_MEAS_ACT_PLANE = 0                    ; Measuring plane is G17
$AC_MEAS_T_NUMBER = 0                     ; No tool has been selected
$AC_MEAS_D_NUMBER = 0
$AC_MEAS_TYPE = 10                        ; Set measuring type on tool length

RETVAL = MEASURE()                        , Start measuring process
if RETVAL <> 0 setal(61000 + RETVAL)
endif
if $AC_MEAS_TOOL_LENGTH <> 10             ; Query known tool length
setal(61000 + $AC_MEAS_TOOL_LENGTH)
endif
m30
```

7.5.5.2 Measurement of tool diameter (measurement type 11)

Tool diameter measurement on a reference part ($AC_MEAS_TYPE = 11)

The tool diameter can be measured on a reference part that has already been measured. Depending on the position of the tool, it is possible to select plane G17 for tool position in the z direction, G18 for tool position in the y direction and G19 for tool position in the x direction.

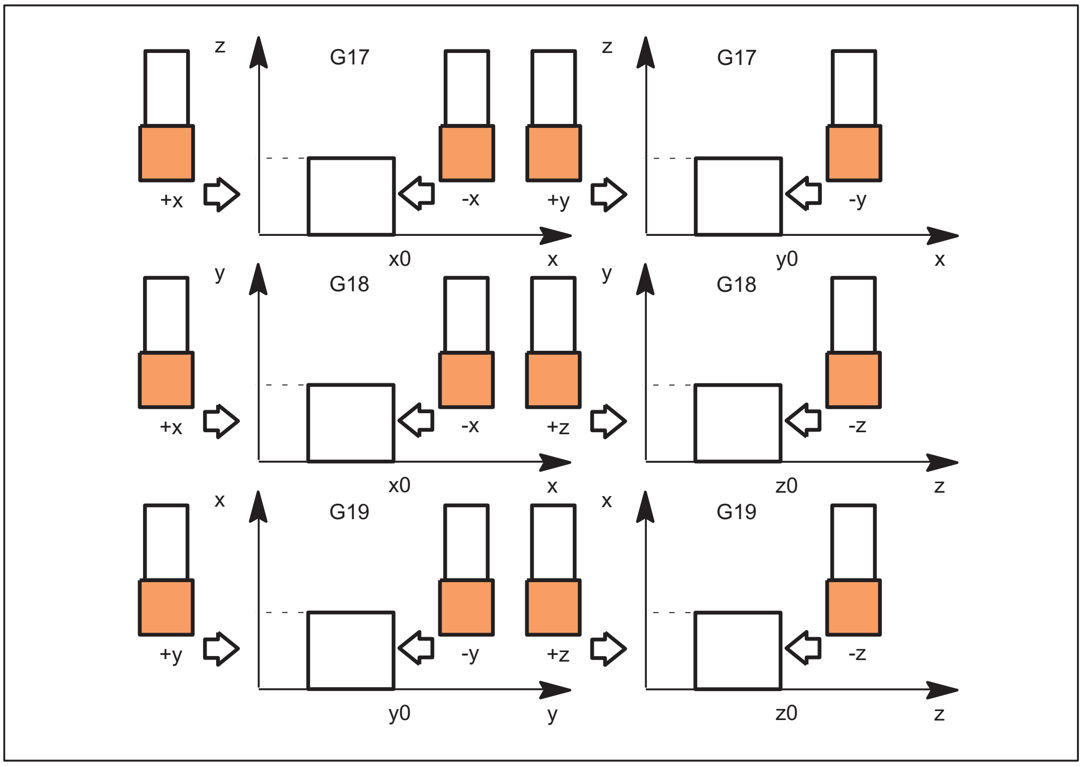


Figure 7-19 Tool diameter for selected planes G17, G18 and G19

The values of the following variables are evaluated for measurement type 11:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_SETPOINT[axis]	Set position x0
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_TYPE	11

* optional

The following output variables are written for measurement type 11:

Output variable	Meaning
$AC_MEAS_TOOL_DIAMETER	Tool diameter

7.5.5.3 Measurement of tool lengths with zoom-in function (measurement type 22)

Tool length with zoom-in function

Tool length measurement with zoom-in function ($AC_MEAS_TYPE = 22)

If a zoom-in function is available on the machine, it can be used to determine the tool dimensions.

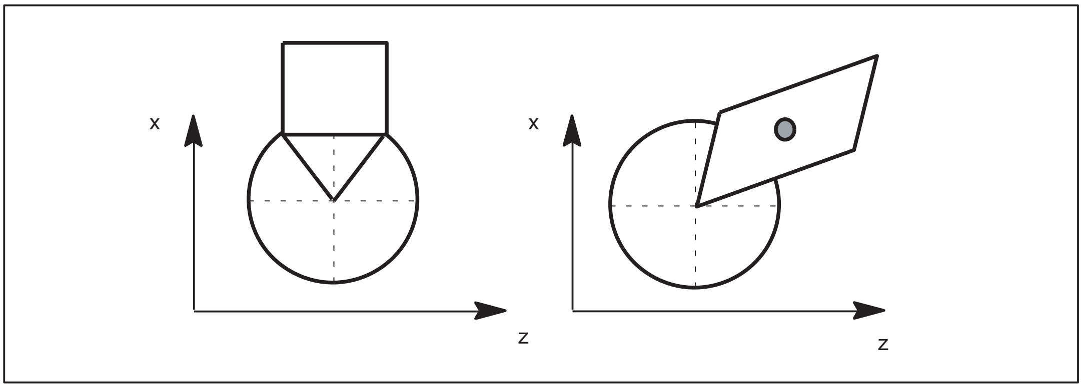


Figure 7-20 Measurement of tool lengths with zoom-in function

The values of the following variables are evaluated for measurement type 22:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for all channel axes
$AC_MEAS_P1_COORD	Coordinate system of the measuring point *
$AA_MEAS_SETPOINT[axis]	Zoom positions x and z must be specified
$AC_MEAS_SET_COORD	Coordinate system of setpoint *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	22

* optional

The following output variables are written for measurement type 22:

Output variable	Description
$AC_MEAS_RESULT[0]	Tool length in x
$AC_MEAS_RESULT[1]	Tool length in y
$AC_MEAS_RESULT[2]	Tool length in z
$AC_MEAS_RESULT[3]	Tool length L1
$AC_MEAS_RESULT[4]	Tool length L2
$AC_MEAS_RESULT[5]	Tool length L3

7.5.5.4 Measuring a tool length with stored or current position (measurement type 23)

Tool length with stored / current position

Tool length measurement with stored or current position ($AC_MEAS_TYPE = 23)

In the case of manual measurement, the tool dimensions can be determined in the X and Z directions. From the known position of the

- Tool carrier reference point and the
- Workpiece dimensions

ShopTurn calculates the tool offset data.

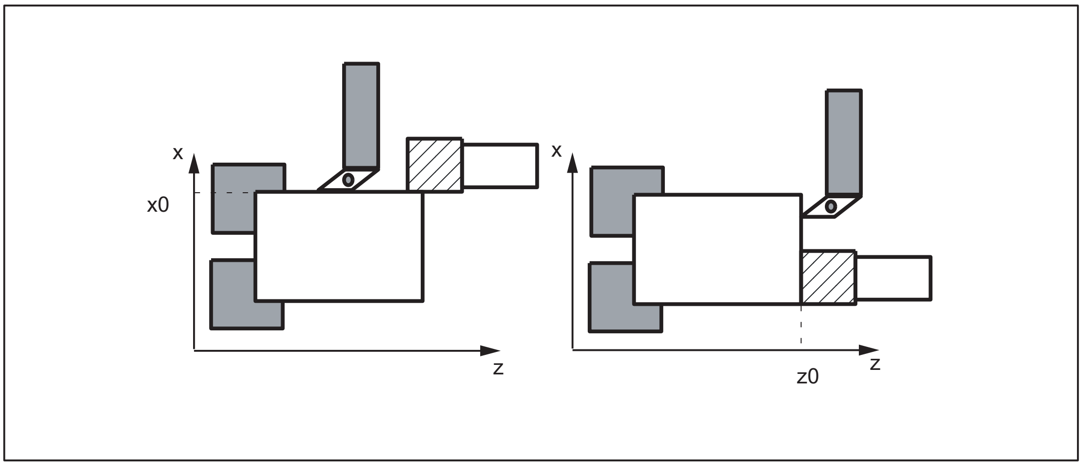


Figure 7-21 Measurement of a tool length with a stored or actual position

The values of the following input variables are evaluated for measurement type 23:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Current or marked position
$AC_MEAS_P1_COORD	Coordinate system of the measuring point *
$AA_MEAS_SETPOINT[axis]	Setpoint position (minimum one geo axis must be specified)
$AC_MEAS_SET_COORD	Coordinate system of setpoint *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TOOL_MASK	Tool position, radius *
$AC_MEAS_DIR_APPROACH	Approach direction *
$AC_MEAS_INPUT[0] = 1	the calculated tool lengths are written to the data management *
$AC_MEAS_TYPE	23

* optional

The following output variables are written for measurement type 23:

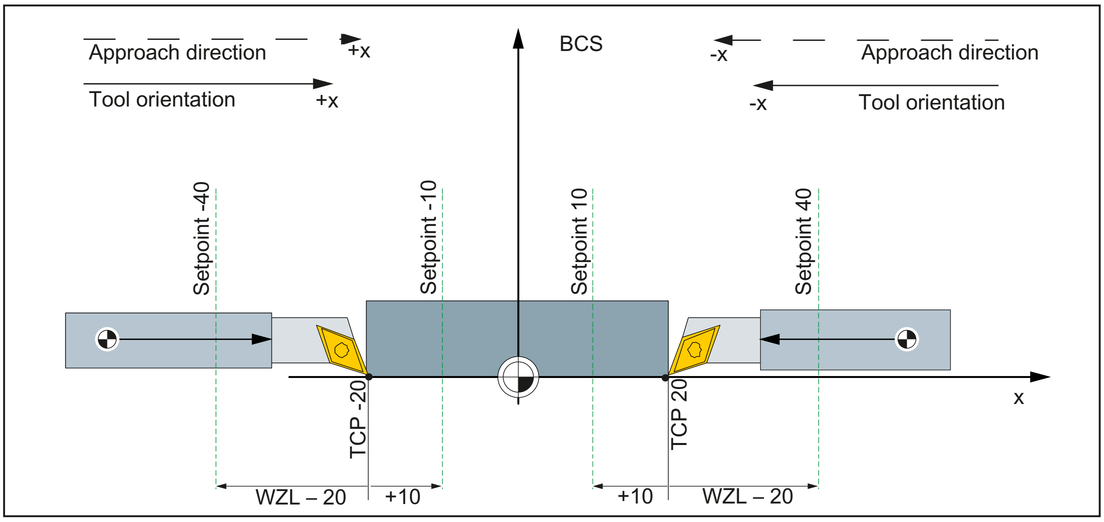
Output variable	Description
$AC_MEAS_RESULT[0]	Tool length in x
$AC_MEAS_RESULT[1]	Tool length in y
$AC_MEAS_RESULT[2]	Tool length in z
$AC_MEAS_RESULT[3]	Tool length L1
$AC_MEAS_RESULT[4]	Tool length L2
$AC_MEAS_RESULT[5]	Tool length L3

7.5.5.5 Measurement of a tool length of two tools with the following orientation:

Tool orientation

For tools whose orientation points to the toolholder shows must be set in the system variables $AC_MEAS_TOOL_MASK, bit 9 = 1 (0x200). The calculated tool lengths are then included negatively.

Two turning tools each with their own reference point with a tool orientation in the approach direction



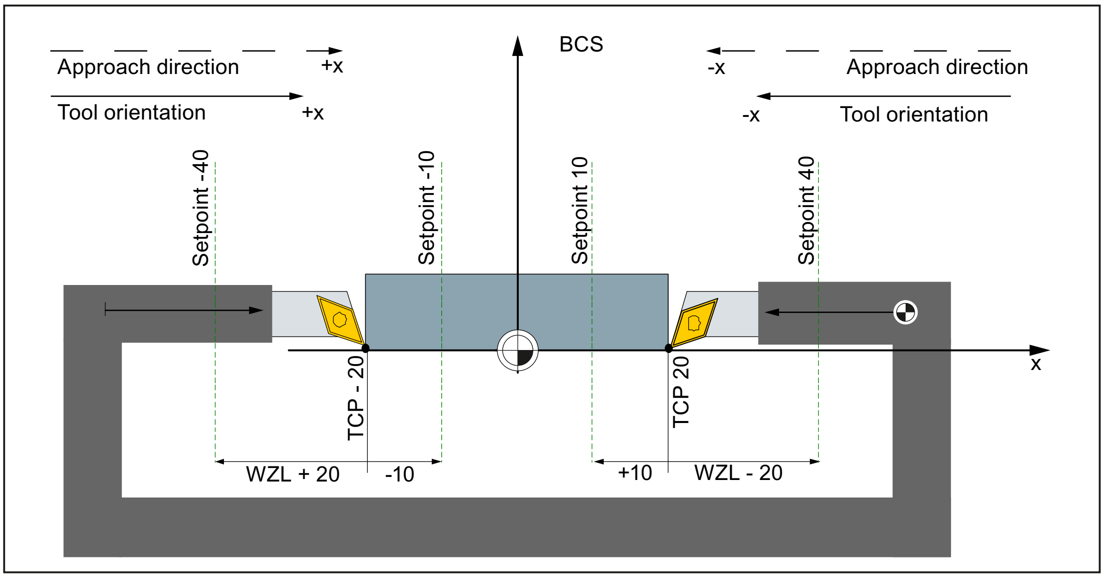
Settings in the system data:

Left-hand tool: Approach direction and tool orientation +x	
System variable	**Meaning**
$AC_MEAS_TOOL_MASK = 0x2	Tool position in x direction (G19)
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction and tool orientation -x	
$AC_MEAS_TOOL_MASK = 0x40	Tool position in the -x direction
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS

Two turning tools each with their own reference point with a tool counter-orientation in the approach direction

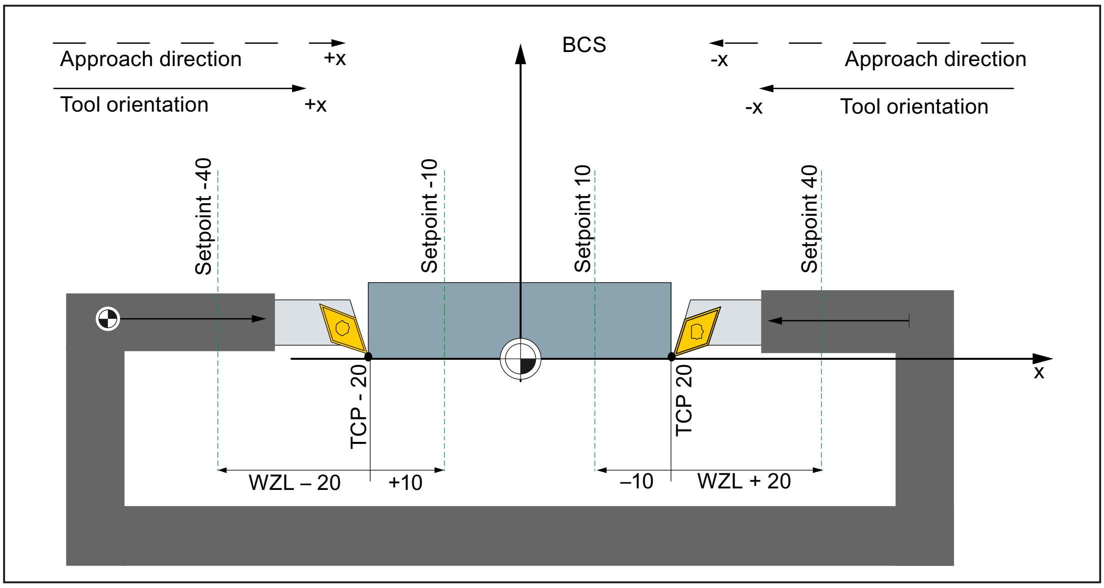


Settings in the system data:

Left-hand tool: Approach direction and tool orientation +x	
System variable	**Meaning**
$AC_MEAS_TOOL_MASK = 0x2 + 0x200	Tool position in x direction (G19) + Tool length differential values are included negatively
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction and tool orientation -x	
$AC_MEAS_TOOL_MASK = 0x40	Tool position in the -x direction
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS



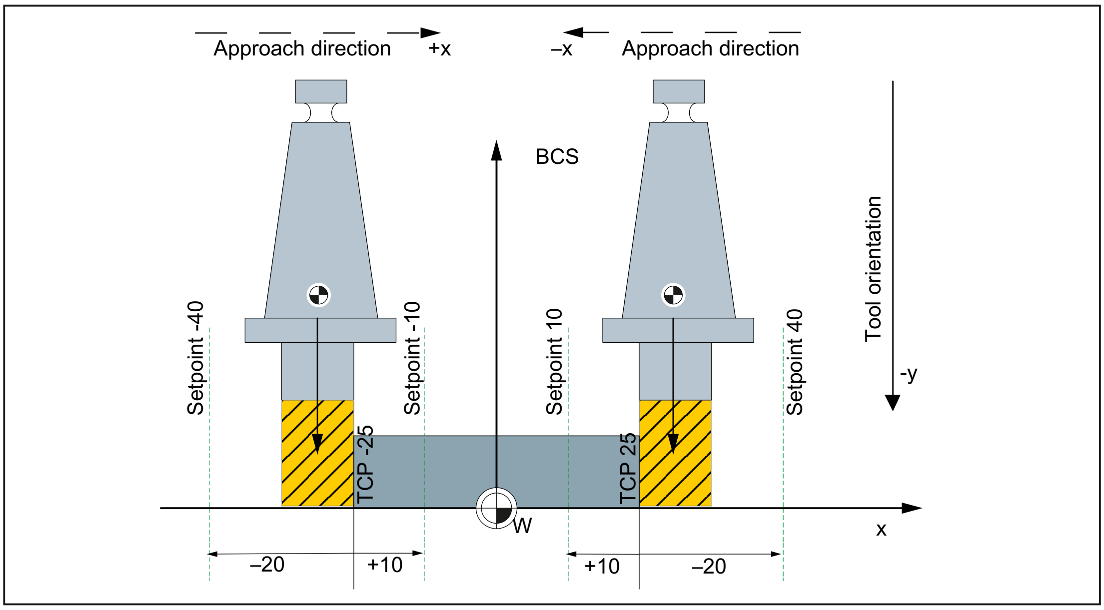
Settings in the system data:

Left-hand tool: Approach direction and tool orientation +x	
System variable	Meaning
$AC_MEAS_TOOL_MASK = 0x2	Tool position in x direction (G19)
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction and tool orientation -x	
$AC_MEAS_TOOL_MASK = 0x40 + 0x200	Tool position in x direction + Tool length differential values are included negatively
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS

Two milling tools each with their own reference point, tool orientation perpendicular to the approach direction



Settings in the system data:

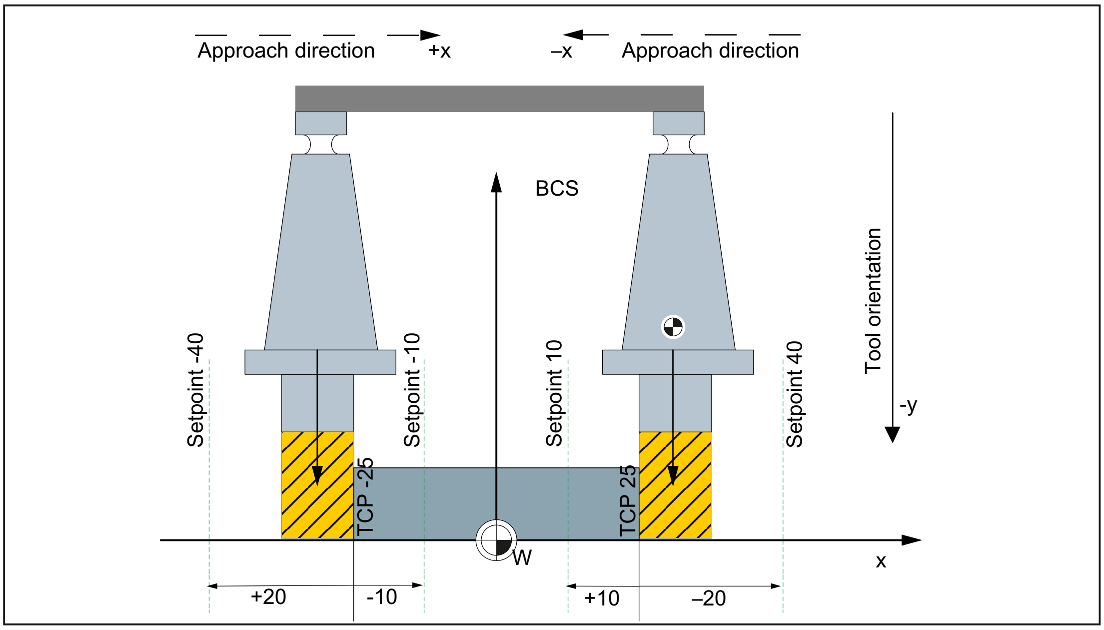
Left-hand tool: Approach direction +x and tool orientation -y	
System variable	**Meaning**
$AC_MEAS_TOOL_MASK = 0x80	Tool position in -y direction
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction -x and tool orientation -y	
$AC_MEAS_TOOL_MASK = 0x80	Tool position in -y direction
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS

Two milling tools each with a reference point, tool orientation perpendicular to the approach direction

Two milling tools with one reference point with a tool orientation in -y

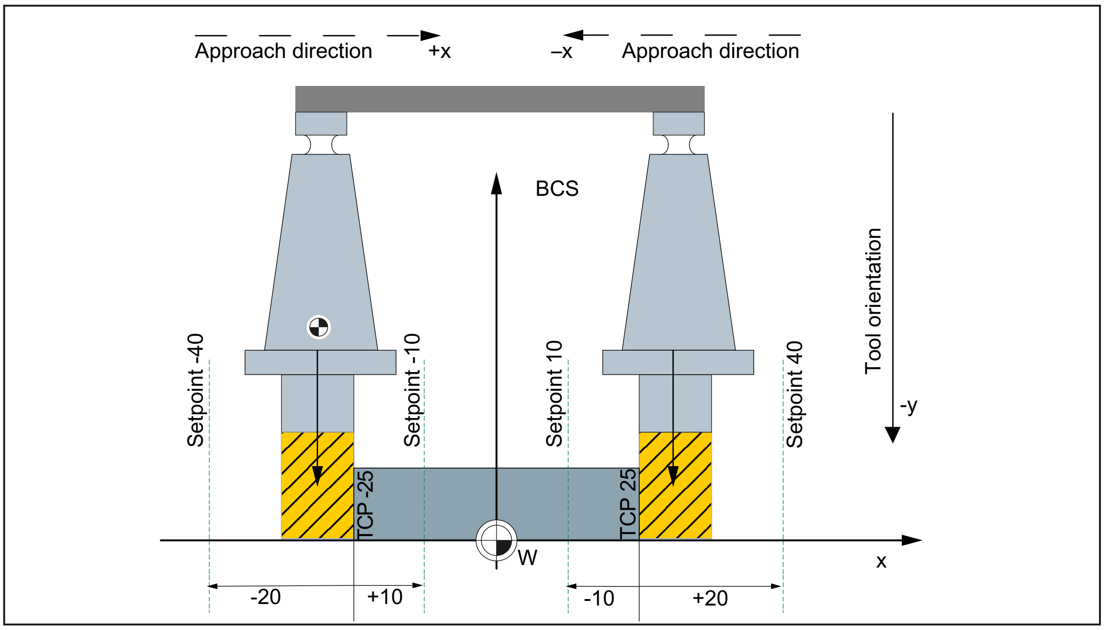


In the present layout, the tool position $AC_MEAS_TOOL_MASK and approach direction to the workpiece $AC_MEAS_DIR_APPROACH must be set as follows:

Left-hand tool: Approach direction +x and tool orientation -y	
System variable	**Meaning**
$AC_MEAS_TOOL_MASK = 0x80 + 0x200	Tool position in -y direction + tool length differential values are included negatively
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction -x and tool orientation -y	
$AC_MEAS_TOOL_MASK = 0x80	Tool position in -y direction
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS



In the present layout, the tool position $AC_MEAS_TOOL_MASK and approach direction to the workpiece $AC_MEAS_DIR_APPROACH must be set as follows:

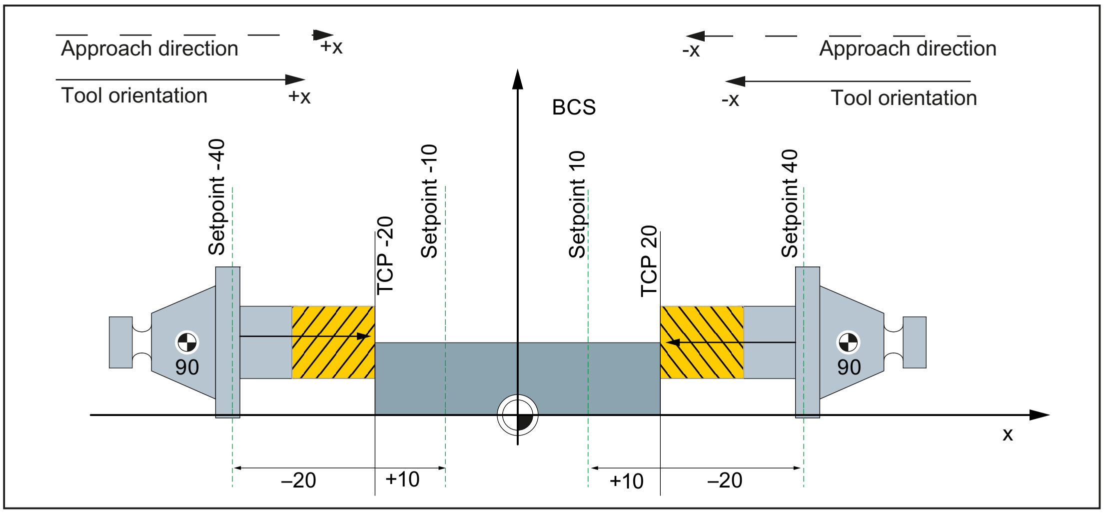
Left-hand tool: Approach direction +x and tool orientation -y	
System variable	**Meaning**
$AC_MEAS_TOOL_MASK = 0x80	Tool position in -y direction
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction -x and tool orientation -y	
$AC_MEAS_TOOL_MASK = 0x80 + 0x200	Tool position in -y direction + tool length differential values are included negatively
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS

Two milling tools each with their own reference point with tool counter-orientation in the approach direction

Two milling tools each with their own reference point with a tool orientation in the approach direction



In the present layout, the tool position $AC_MEAS_TOOL_MASK and approach direction to the workpiece $AC_MEAS_DIR_APPROACH must be set as follows:

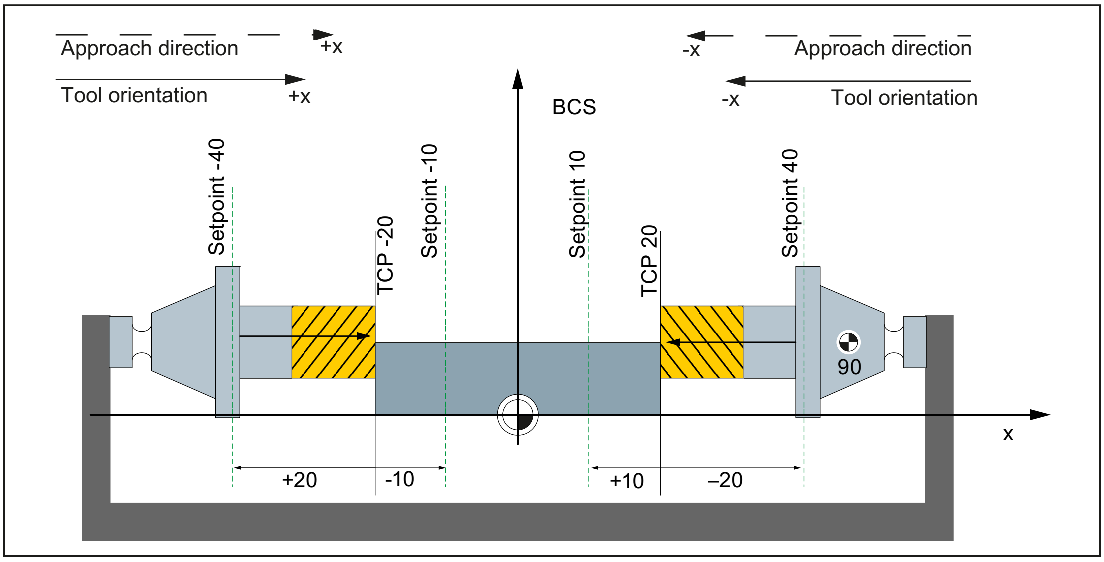
Left-hand tool: Approach direction and tool orientation +x	
System variable	Meaning
$AC_MEAS_TOOL_MASK = 0x2	Tool position in x direction (G19)
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction and tool orientation -x	
$AC_MEAS_TOOL_MASK = 0x40	Tool position in the -x direction
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS

Two milling tools each with a reference point with tool counter-orientation in the approach direction

Two milling tools with one reference point with a tool position opposite to the orientation

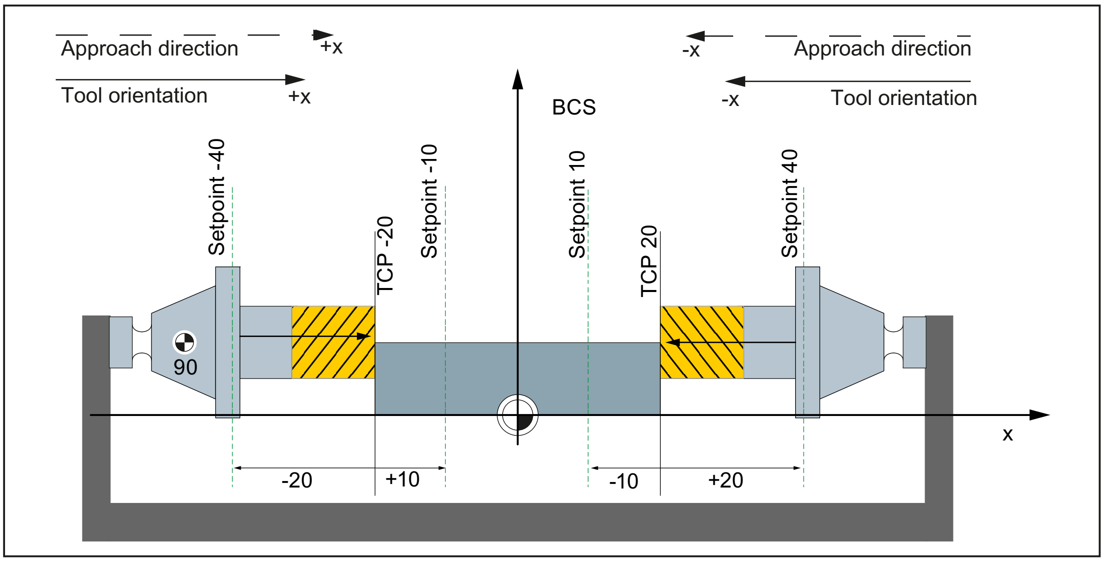


In the present layout, the tool position $AC_MEAS_TOOL_MASK and approach direction to the workpiece $AC_MEAS_DIR_APPROACH must be set as follows:

Left-hand tool: Approach direction and tool orientation +x	
System variable	**Meaning**
$AC_MEAS_TOOL_MASK = 0x2 + 0x200	Tool position in x direction (G19) + tool length differential values are included negatively
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction and tool orientation -x	
$AC_MEAS_TOOL_MASK = 0x40	Tool position in the -x direction
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS



In the present layout, the tool position $AC_MEAS_TOOL_MASK and approach direction to the workpiece $AC_MEAS_DIR_APPROACH must be set as follows:

Left-hand tool: Approach direction and tool orientation +x	
System variable	Meaning
$AC_MEAS_TOOL_MASK = 0x2	Tool position in x direction (G19)
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction and tool orientation -x	
$AC_MEAS_TOOL_MASK = 0x40 + 0x200	Tool position in -x direction + tool length differential values are included negatively
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 1	Coordinate system of x-th measuring point = BCS
$AC_MEAS_SET_COORD = 1	Coordinate system of the setpoint = BCS

Different tools in the WCS

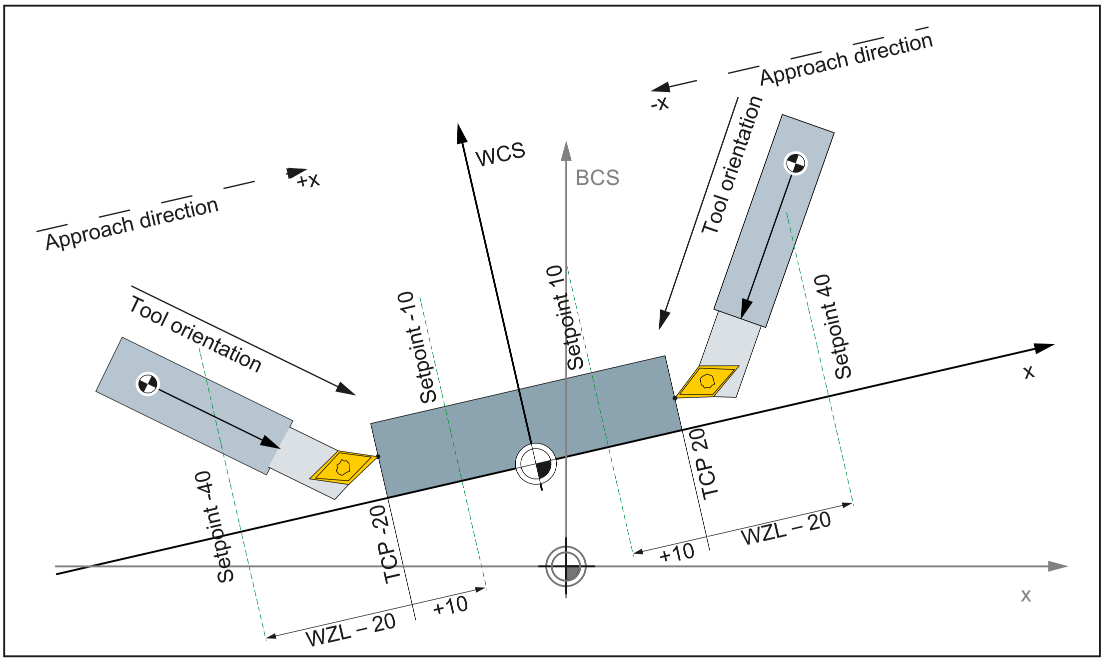


Figure 7-22 Two turning tools each with their own reference point

Settings in the system data:

Left-hand tool: Approach direction +x and tool orientation -y	
System variable	**Meaning**
$AC_MEAS_TOOL_MASK = 0x0	All tool lengths are considered (default setting)
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction -x and tool orientation -y	
$AC_MEAS_TOOL_MASK = 0x0	All tool lengths are considered (default setting)
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 0	Coordinate system of x-th measuring point = WCS (default setting)
$AC_MEAS_SET_COORD = 0	Coordinate system of the setpoint = WCS (default setting)

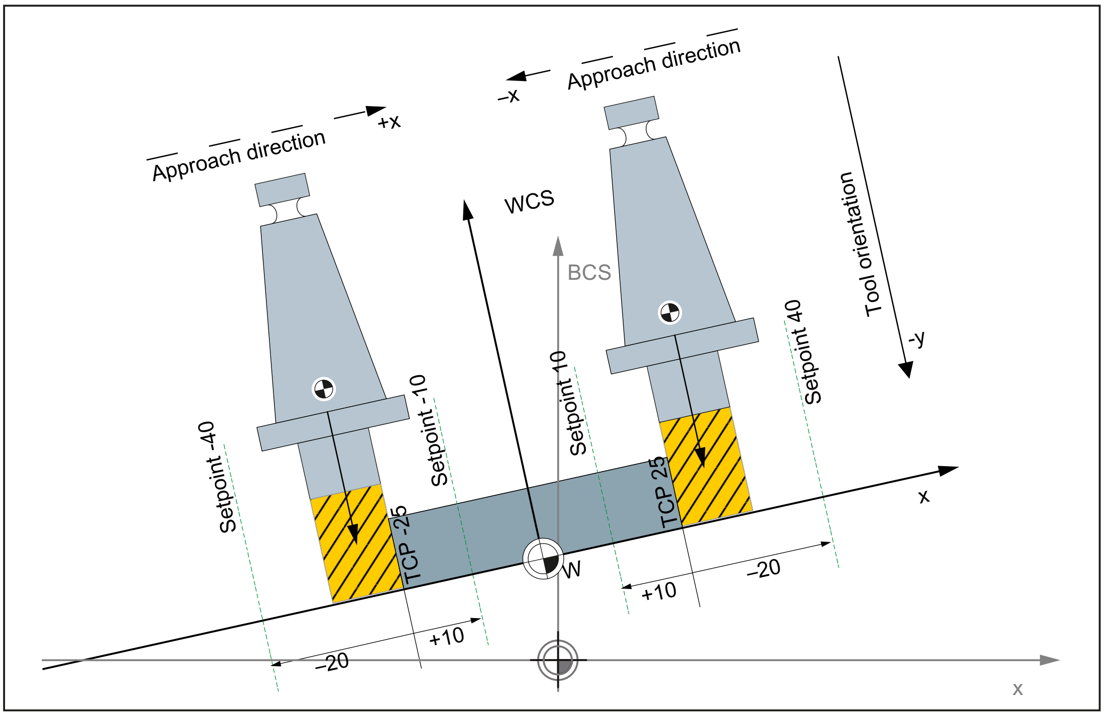


Figure 7-23 Two milling tools each with its own reference point

Settings in the system data:

Left-hand tool: Approach direction +x and tool orientation -y	
System variable	Meaning
$AC_MEAS_TOOL_MASK = 0x80	Tool position in -y direction
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction -x and tool orientation -y	
$AC_MEAS_TOOL_MASK = 0x80	Tool position in -y direction
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 0	Coordinate system of x-th measuring point = WCS (default setting)
$AC_MEAS_SET_COORD = 0	Coordinate system of the setpoint = WCS (default setting)

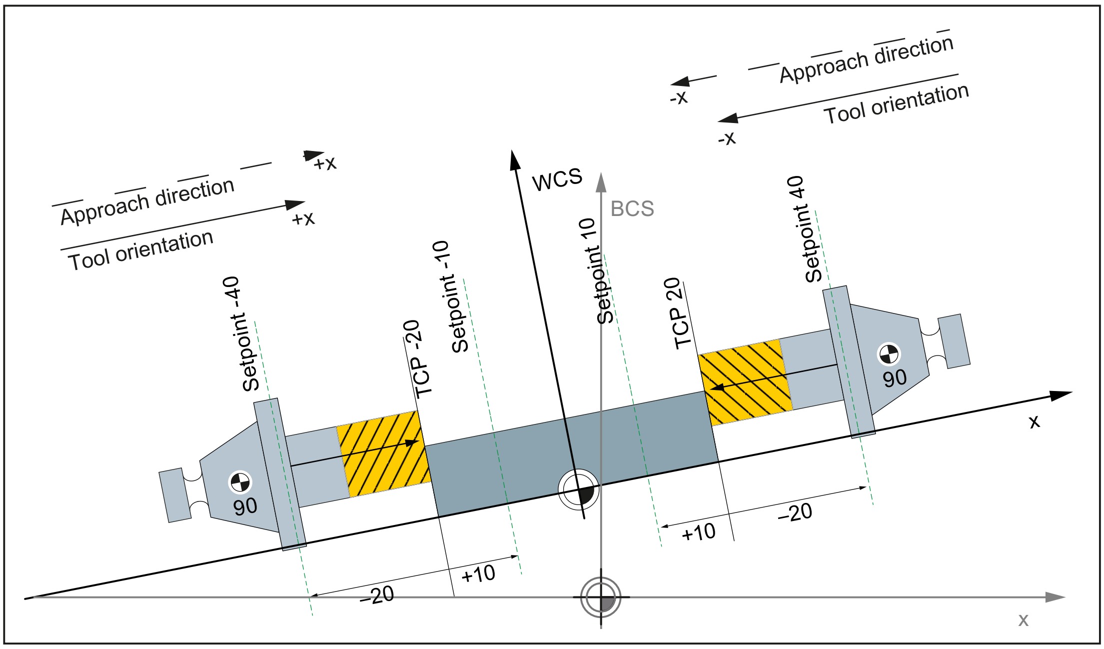


Figure 7-24 Two milling tools rotated at 90 degrees each with their own reference point

Settings in the system data:

Left-hand tool: Approach direction +x and tool orientation -y	
System variable	Meaning
$AC_MEAS_TOOL_MASK = 0x2	Tool position in x direction (G19)
$AC_MEAS_DIR_APPROACH = 0	Approach direction +x

Right-hand tool: Approach direction -x and tool orientation -y	
$AC_MEAS_TOOL_MASK = 0x40	Tool position in the -x direction
$AC_MEAS_DIR_APPROACH = 1	Approach direction -x

For both tools	
$AC_MEAS_Px_COORD = 0	Coordinate system of x-th measuring point = WCS (default setting)
$AC_MEAS_SET_COORD = 0	Coordinate system of the setpoint = WCS (default setting)

7.6 Measurement accuracy and functional testing

7.6.1 Measurement accuracy

The measuring accuracy is affected by the following parameters:

- Delay time of the measuring signal (T_{Delay})
- Traversal speed during the measurement (v_M)

Delay time compensation of the measuring signal (T_{Delay})

The delay time of the measuring signal, i.e. the time from the initiation of the probe until the saving of the measured value in the control depends on the response time of the probe and the signal runtime of the control hardware. The control compensates for the delay time during the measurement. This requires that the delay time is determined and entered in the following machine data:

MD13220 $MN_MEAS_PROBE_DELAY_TIME = <determined delay time>

Note

Maximum compensated delay time $T_{MaxDelay}$

$T_{MaxDelay}$ = 15 * position controller or DP cycle

The compensation of a delay time > $T_{MaxDelay}$ is not sensible from the control viewpoint. This means larger values are limited to $T_{MaxDelay}$.

Maximum traversal speed during the measurement (v_M)

The maximum permitted traversal speed for a measurement depends on the number of programmed measuring edges and the parameterized position-controller or DP cycle.

To receive correct results, the traversal speed during the measurement must be chosen so that the following conditions are satisfied every **two** position-controller or DP cycles:

- Maximum **one identical** trigger signal, i.e. one positive **or** one negative edge of **one** probe
- Maximum **four different identical** trigger signals, i.e. one positive and one negative edge of **two** probes

7.6.2 Probe function test

Example of function test

Table 7- 7

Program code	Comment
%_N_PRUEF_MESSTASTER_MPF	
;$PATH=/_N_MPF_DIR	
;Testing program probe connection	
N05 DEF INT MTSIGNAL	;Flag for trigger status
N10 DEF INT ME_NR=1	; measurement input number
N20 DEF REAL MESSWERT_IN_X	
N30 G17 T1 D1	; tool compensation for
	; preselect probe
N40 _ANF: G0 G90 X0 F150	; Starting position and
	; measuring velocity
N50 MEAS=ME_NR G1 X100	; measurement at measurement input =1
	; in the X axis
N60 STOPRE	
N70 MTSIGNAL=$AC_MEA[1]	; read software switching signal
	; at 1st measurement input
N80 IF MTSIGNAL == 0 GOTOF _FEHL1	; evaluation of signal
N90 MESSWERT_IN_X=$AA_MW[X]	; Read in measured value of
	; workpiece coordinates
N95 M0	
N100 M02	
N110 _FEHL1: MSG ("Probe not switching!")	
N120 M0	
N130 M02	

7.7 Simulated measuring

7.7.1 General functionality

Brief description

To make measurements at real machines, probes must be connected which supply switching signals at certain positions. Probes are not used when making measurements in simulated environments - the switching positions are specified in a different way.

Simulated measuring supports two ways of entering switching positions:

- Position-related switch request: The switching position is derived from the axial end position programmed in the measuring block.
- External switching request: The switching position is defined by controlling a digital output.

Preconditions

For simulated measuring, all of the machine axes in the system must be parameterized as simulated axes:

- MD30130 $MA_CTRLOUT_TYPE[axis] = 0 (simulated setpoint)
- MD30240 $MA_ENC_TYPE[axis] = 0 (simulated encoder)

7.7.2 Position-related switch request

Function

"Position-related switch request" is selected using the following NCK-specific machine data:

- MD13230 $MN_MEAS_PROBE_SOURCE = 0
- MD13231 $MN_MEAS_PROBE_OFFSET = <position offset>

The axial switching position is calculated from the axial end position programmed in the measuring block and the parameterized position offset:

Switching position[axis] = End position[axis] - position offset

During the measuring block, it is cyclically checked as to whether the switching position of the axis is reached:

Setpoint position[axis] ≥ switching position[axis]

When the switching position is reached, the rising edge of the switching signal is generated for probes 1 and 2. One position controller cycle later, the following edges.

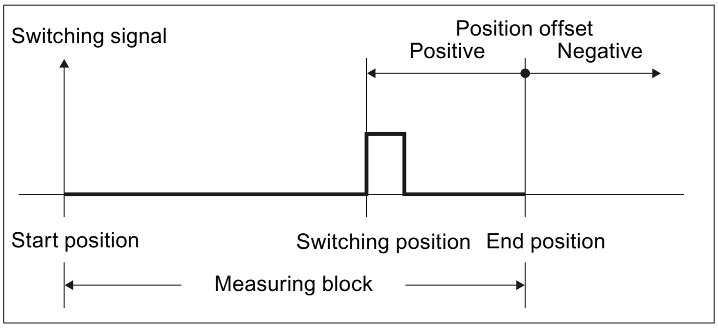


Figure 7-25 Position-dependent switch request

The measured value is the actual value of the axis at the instant in time that the switching signal programmed in the measuring block occurs (rising / falling edge).

If several axes are programmed in a measuring block, then a dedicated switching position is obtained for each axis by the position offset that is axially taken into consideration. The probe signal is generated at the first axial switching position that is reached.

Note

Probe signals

The probe signals are always simultaneously generated for probes 1 and 2.

Negative offset values

The switching position is shifted behind the end position by entering a negative value for the position offset. In this case, no probe signals are generated.

Examples

The position offset is set to 0.1 mm: MD13231 $MN_MEAS_PROBE_OFFSET = 0.1

Example 1: Channel-specific measuring in 2 axes

Program code	Comment
N10 G01 G90	
N20 MEAS=1 X100 Y10 F100	; rising edge, probe 1
	; Switching position[X] = 99.9
	; Switching position[Y] = 9.9

Example 2: Axial measuring using synchronized action

Program code	Comment
N10 G01 G90	
N15 WHEN TRUE DO MEASA[X]=(1,1)	; rising edge, probe 1
N20 X10 F100	; Switching position[X] = 9.9

7.7.3 External switch request

Function

The "external switching request" is selected using the NCK specific machine data by entering the number (1...8) of the digital output being used:

- MD13230 $MN_MEAS_PROBE_SOURCE = <number of the digital output>

The probe signal is triggered by controlling the configured digital output. It is not necessary to hard-wire the digital output to a measuring input.

The rising edge of the switching signal for probes 1 and 2 is generated by setting the digital output. The falling edges are generated by resetting the digital output.

The measured value is the actual value of the axis at the instant in time that the switching signal programmed in the measuring block occurs (rising / falling edge).

Digital output: Configuration

The following machine data must be set to be able to use digital outputs for simulated measuring:

- MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS = 1 (number of active digital NCK output bytes)
- MD13120 $MN_CONTROL_UNIT_LOGIC_ADDRESS = 0 (logical address, SINAMICS-CU)

Digital output: Setting

The configured digital output can be set in a synchronized action:

```
WHEN <condition> DO $A_OUT[<number of digital output>] = 1
```

Examples

Digital output used: MD13230 $MN_MEAS_PROBE_SOURCE = 1

Example 1: Channel-specific measuring in 2 axes

Program code	Comment
N10 G01 G90 $A_OUT[1]=0	; Preset digital output 1
N15 WHEN $AC_DETW<=10 DO $A_OUT[1]=1	; Path residual distance <= 10 => Dig. output 1 = 1
N20 MEAS=1 X100 Y10 F100	; rising edge, probe 1

Example 2: Axial measurement

Program code	Comment
N10 G01 G90 $A_OUT[1]=0	; Preset digital output 1
N15 WHEN $AA_IW[X]>=80 DO $A_OUT[1]=1	; Axial setpoint >= 80 => Dig. output 1 = 1
N20 MEASA[X]=(1,1) X100 F100	; rising edge, probe 1

7.7.4 System variable

For simulated measuring, the following system variables have the same functionality as for real measuring:

- $AC_MEA (probe has responded)
- $AA_MEAACT (axial measuring active)
- $AA_MM (acquired probe position (MCS))
- $AA_MM1...4 (probe position 1st – 4th trigger (MCS))
- $AA_MW (acquired probe position (WCS))
- $AA_MW1...4 (probe position 1st trigger (WCS))

The following system variable does not supply sensible values:

- $A_PROBE (probe state)

7.8 Channels - only 840D sl

7.8.1 Measuring mode 1

Supplementary conditions

- One-time measurement
- One probe
- Trigger signals are the rising and falling edges

Measurement with one encoder - actual value for the current encoder

Program code

```
N2    MEASA[X] = (1, 1, -1) G01 X100 F100
N3    STOPRE
N4    IF $AC_MEA[1]==FALSE gotof ENDE
N5    R10=$AA_MM1[X]
N6    R11=$AA_MM2[X]
N7    END
```

Measurement with two encoders - actual values for two encoders

Program code
N2 MEASA[X] = (31, 1, -1) G01 X100 F100
N3 STOPRE
N4 IF $AC_MEA[1]==FALSE gotof ENDE
N5 R10=$AA_MM1[X]
N6 R11=$AA_MM2[X]
N7 R12=$AA_MM3[X]
N8 R13=$AA_MM4[X]
N9 END

7.8.2 Measuring mode 2

Supplementary conditions

- Two probes
- Trigger signals are the rising and falling edges
- Actual value from the current encoder

Program code
N2 MEASA[X] = (2, 1, -1, 2, -2) G01 X100 F100
N3 STOPRE
N4 IF $AC_MEA[1]==FALSE gotof MESSTASTER2
N5 R10=$AA_MM1[X]
N6 R11=$AA_MM2[X]
N7 PROBE2
N8 IF $AC_MEA[2]==FALSE gotof ENDE
N9 R12=$AA_MM3[X]
N10 R13=$AA_MM4[X]
N11 END:

7.8.3 Continuous measurement

Supplementary conditions

- The measurement is done in measuring mode 1:
- Measurement with 100 values
- One probe
- Trigger signal is the falling edge
- Actual value from the current encoder

Continuous measurement on completion of programmed traversing movement

Program code		Comment
N1	DEF REAL MESSWERT[100]	
N2	DEF INT INDEX=0	
N3	MEAC[x]=(1, 1, -1) G01 X1000 F100	
N4	MEAC[X]=(0)	; Abort
N5	R1=$AC_FIFO1[4]	;Number of measured values
N6	FOR INDEX=0 TO R1	
N7	MESSWERT[INDEX]=$AC_FIFO1[0]	; Read out measured values
N8	ENDFOR:	

Continuous measurement with deletion of distance-to-go

Delete distance-to-go after last measurement.

Program code		Comment
N1	DEF INT ANZAHL=100	
N2	DEF REAL MESSWERT[ANZAHL]	
N3	DEF INT INDEX=0	
N4	WHEN $AC_FIFO1[4]==ANZAHL DO DELDTG (X) MEAC[X] =(0)	
N5	MEAC[X]=(1, 1, -1) G01 X1000 F100	; Start measurement
N6	R1=$AC_FIFO1[4]	;Number of measured values
N7	FOR INDEX=0 TO R1	
N8	MESSWERT[INDEX]=$AC_FIFO1[0]	; Read out measured values
N9	ENDFOR:	

Continuous modal measurement over several blocks

Program code		Comment
N1	DEF INT ANZAHL=100	
N2	DEF REAL MESSWERT[ANZAHL]	
N3	DEF INT INDEX=0	
N4	ID=1 MEAC[X]=(1, 1, -1)	; Start measurement
N5	ID=2 WHEN $AC_FIFO1[4]==ANZAHL DO MEAC[X]=(0) CANCEL(2)	
N6	G01 X1000 Y100	
N7	X100 Y100	
N8	R1=$AC_FIFO1[4]	;Number of measured values
N9	FOR INDEX=0 TO R1	
N10	MESSWERT[INDEX]=$AC_FIFO1[0]	; Read out measured values
N11	ENDFOR:	

7.8.4 Functional test and repeat accuracy

Function test

Program code	Comment
%_N_PRUEF_MESSTASTER_MPF	
;$PATH=/_N_MPF_DIR	
;Testing program probe connection	
N05 DEF INT MTSIGNAL	;Flag for trigger status
N10 DEF INT ME_NR=1	; measurement input number
N20 DEF REAL MESSWERT_IN_X	
N30 G17 T1 D1	; tool compensation for
	; preselect probe
N40 _ANF: G0 G90 X0 F150	; Starting position and
	; measuring velocity
N50 MEAS=ME_NR G1 X100	; measurement at measurement input =1
	; in the X axis
N60 STOPRE	
N70 MTSIGNAL=$AC_MEA[1]	; read software switching signal
	; at 1st measurement input
N80 IF MTSIGNAL == 0 GOTOF _FEHL1	; evaluation of signal
N90 MESSWERT_IN_X=$AA_MW[X]	; Read in measured value of
	; workpiece coordinates
N95 M0	
N100 M02	
N110 _FEHL1: MSG ("Probe not switching!")	
N120 M0	
N130 M02	

Repeat accuracy

This program allows the measuring scatter (repeat accuracy) of the entire measuring system (machine-probe-signal transmission to NC) to be calculated.

In the example, ten measurements are taken in the X axis and the measured value recorded in the workpiece coordinates.

It is therefore possible to determine the random dimensional deviations which are not subject to any trend.

Program code	Comment
%_N_PRUEF_GENAU_MPF;	
$PATH=/_N_MPF_DIR	
N05 DEF INT SIGNAL, II	; Variable definition

Program code	Comment
N10 DEF REAL MESSWERT_IN_X[10]	
N15 G17 T1 D1	; Initial conditions,
	: Tool compensation
	; preselect for probe
N20 _ANF: G0 X0 F150 ←	; Prepositioning in the measured axis
N25 MEAS=+1 G1 X100 ←	; at 1st measurement input when
	; switching signal not deflected,
	; deflected in the X axis
N30 STOPRE ←	; Stop decoding for this after
	; subsequent evaluation of
	; results
N35 SIGNAL= $AC_MEA[1]	; read software switching signal at
	; 1. Read measuring input
N37 IF SIGNAL == 0 GOTOF_FEHL1	; Check switching signal
N40 MESSWERT_IN_X[II]=$AA_MW[X]	; Read measured value in workpiece coordinates
N50 II=II+1	
N60 IF II<10 GOTOB_ANF	; Repeat 10 times
N65 M0	
N70 M02	
N80 _FEHL1: MSG ("Probe not switching")	
N90 M0	
N95 M02	

After the parameter display (user-defined variables) has been selected, the measurement results can be read in field MEASVALUE_IN_X[10] provided that the program is still being processed.

7.9 Data lists

7.9.1 Machine data

7.9.1.1 General machine data

Number	Identifier: $MN_	Meaning
13200	MEAS_PROBE_LOW_ACTIVE	Switching characteristics of probe
13201	MEAS_PROBE_SOURCE	Measuring pulse simulation via digital output
13210	MEAS_TYPE	Type of measurement for PROFIBUS DP drives
13211	MEAS_CENTRAL_SOURCE	Central measuring data source with PROFIBUS DP drives

7.9.1.2 Channel-specific machine data

Number	Identifier: $MC_	Meaning
20360	TOOL_PARAMETER_DEF_MASK	Definition of tool parameters
28264	LEN_AC_FIFO	Length of $AC_FIFO ... FIFO variables

7.9.2 System variables

Table of all the input values

Identifier	Meaning
$AC_FIFO1…10	FIFO variable 1 to 10
$AC_MEAS_SEMA	Interface assignment
$AC_MEAS_VALID	Validity bits for input values
$AA_MEAS_POINT1	1. Measuring point for all channel axes
$AA_MEAS_POINT2	2. Measuring point for all channel axes
$AA_MEAS_POINT3	3. Measuring point for all channel axes
$AA_MEAS_POINT4	4. Measuring point for all channel axes
$AA_MEAS_SETPOINT	Setpoint position for all channel axes
$AA_MEAS_SETANGLE	Setpoint angle for all channel axes
$AC_MEAS_P1_COORD	Coord. system for the 1st measuring point
$AC_MEAS_P2_COORD	Coord. system for the 2nd measuring point
$AC_MEAS_P3_COORD	Coord. system for the 3rd measuring point
$AC_MEAS_P4_COORD	Coord. system for the 4th measuring point
$AC_MEAS_SET_COORD	Coordinate system of the setpoint
$AC_MEAS_LATCH[0...3]	Pick up measuring points in the WCS
$AA_MEAS_P1_VALID	1. Pick up measuring point in the WCS
$AA_MEAS_P2_VALID	2. Pick up measuring point in the WCS
$AA_MEAS_P3_VALID	3. Pick up measuring point in the WCS
$AA_MEAS_P4_VALID	4. Pick up measuring point in the WCS
$AA_MEAS_SP_VALID	Set setpoint position of axis as valid
$AC_MEAS_WP_SETANGLE	Setpoint workpiece position angle
$AC_MEAS_CORNER_SETANGLE	Setpoint cutting angle of corner
$AC_MEAS_DIR_APPROACH	Approach direction
$AC_MEAS_ACT_PLANE	Working plane for the workpiece
$AC_MEAS_SCALEUNIT	Unit of measurement INCH / METRIC
$AC_MEAS_FINE_TRANS	Corrections in fine displacement
$AC_MEAS_FRAME_SELECT	Frame selection for the workpiece measurement

Identifier	Meaning
$AC_MEAS_CHSFR	Frame chain setting: System frames
$AC_MEAS_NCBFR	Frame chain setting: Global basic frames
$AC_MEAS_CHBFR	Frame chain setting: Channel basic frames
$AC_MEAS_UIFR	Frame chain setting: Settable frames
$AC_MEAS_PFRAME	Frame chain setting: Program frame
$AC_MEAS_T_NUMBER	Tool selection
$AC_MEAS_D_NUMBER	Cutting edge selection
$AC_MEAS_TOOL_MASK	Tool settings
$AC_MEAS_TYPE	Measuring type
$AC_MEAS_INPUT	Measurement input parameters

Table of all the output values

Identifier	Meaning
$A_PROBE[1,2]	Probe status
$A_PROBE_LIMITED[1,2]	Measuring velocity exceeded
$AC_MEA[1,2]	Probe has responded
$AA_MM	Acquired probe position (MCS)
$AA_MM1...4	Probe position 1st to 4th trigger event (MCS)
$AA_MW	Acquired probe position (WCS)
$AA_MW1...4	Probe position 1st to 4th trigger event (WCS)
$AC_MEAS_FRAME	Result frame
$AC_MEAS_WP_ANGLE	Calculated workpiece position angle
$AC_MEAS_CORNER_ANGLE	Calculated angle of intersection
$AC_MEAS_DIAMETER	Calculated diameter
$AC_MEAS_TOOL_LENGTH	Calculated tool length
$AC_MEAS_RESULTS	Measurement results (depending on measurement type)

N3: Software cams, position switching cycles - only 840D sl

8

8.1 Brief Description

Function

The "Software cams" function generates position-dependent switching signals for axes that supply an actual position value (machine axes) and for simulated axes.
These cam signals can be output to the PLC and also to the NCK I/Os.

The cam positions at which signal outputs are set can be defined and altered via setting data. The setting data can be read and written via HMI, PLC and part program.

Activation

The "Software cams" function can be activated and used in all operating modes. The function remains active in the event of reset or Emergency Stop.

Field of application

Output cam signals can be used, for example:

- To activate protection zones
- To initiate additional movements as a function of position
- As reversing signals for hydraulically controlled oscillation axes

Axis types

Software cams can be used on linear and modulo rotary axes that are defined as machine axes.

Cam range/cam pair

Cams are always assigned in pairs to axes. A pair consists of a plus and a minus cam. 32 cam pairs are available.

The plus and minus cams each simulate a mechanical cam which is actuated at a defined point (cam position) in a specific approach direction when the axis reaches the cam position.

Cam ranges are assigned to the plus and minus cams as follows:

- Cam range plus: All positions ≥ plus cam
- Cam range minus: All positions ≤ minus cam

8.2 Cam signals and cam positions

8.2.1 Generation of cam signals for separate output

Separate output of the plus and minus cam signals makes it easy to detect whether the axis is within or outside the plus or minus cam range.

Linear axes

The switching edges of the cam signals are generated as a function of the axis traversing direction:

- The minus cam signal switches from 1 to 0 when the axis traverses the minus cam in the positive axis direction.
- The plus cam signal switches from 0 to 1 when the axis traverses the plus cam in the positive direction.

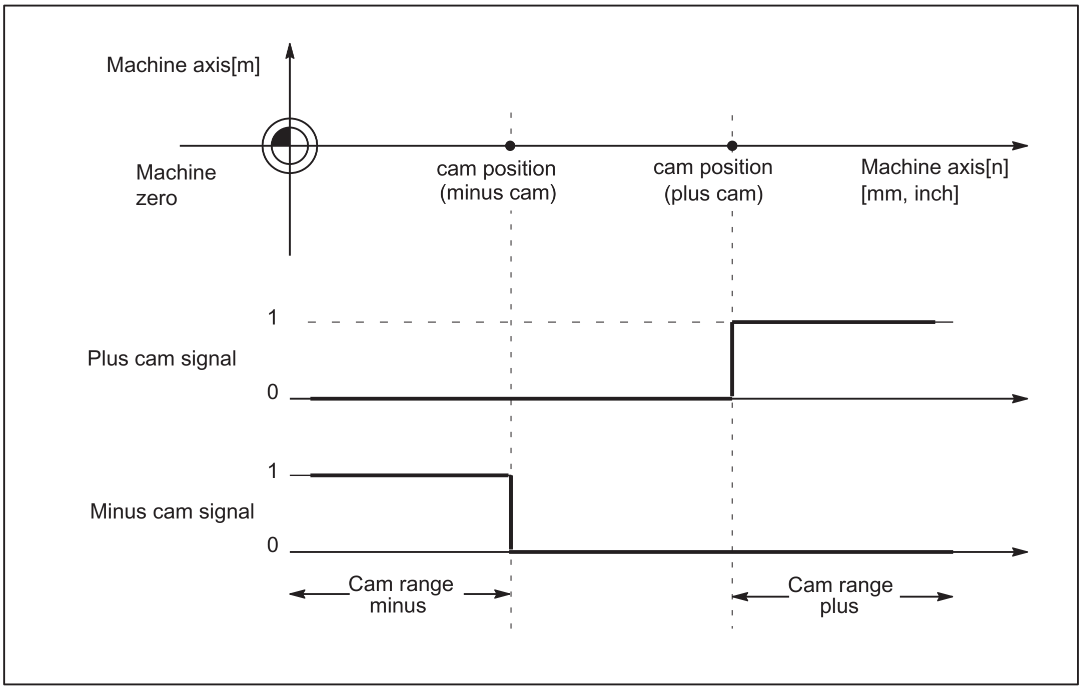


Figure 8-1 Software cams for linear axis (minus cam < plus cam)

Note

If the axis is positioned exactly at the output cam position (plus or minus), the defined output flickers. If the axis moves one increment further, the output becomes a definite zero or one.

Flickering of the actual position causes the signals to flicker in this manner.
The actual position is evaluated.

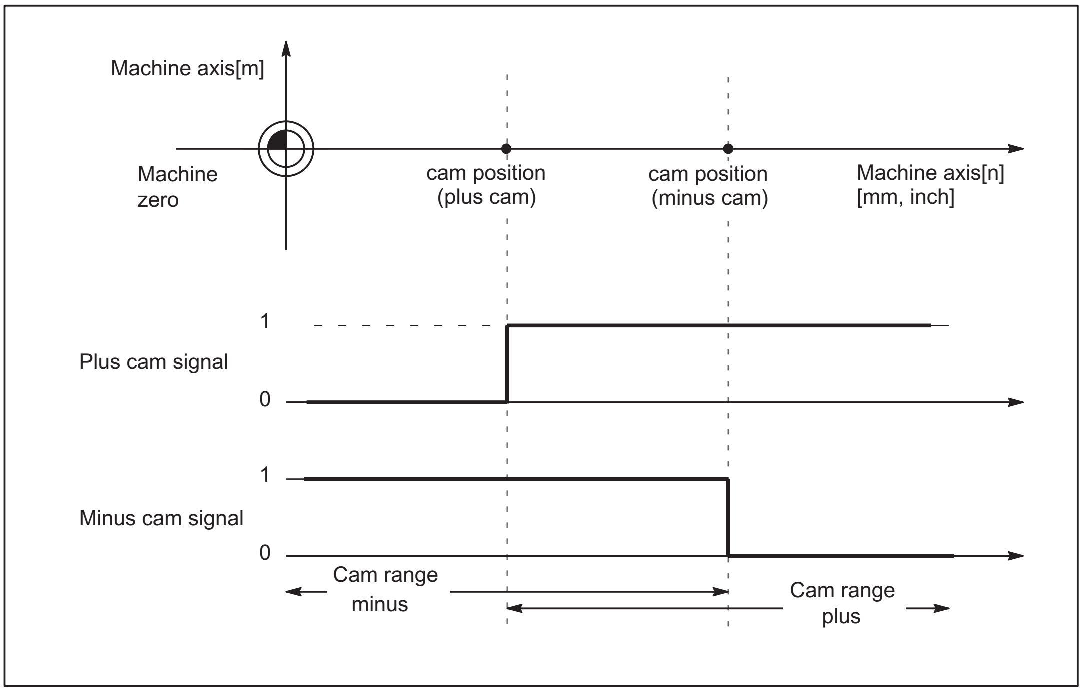


Figure 8-2 Software cams for linear axis (plus cam < minus cam)

Modulo rotary axes

The switching edges of the cam signals are generated as a function of the rotary axis traversing direction:

- The plus cam signal switches from 0 to 1 when the axis traverses the minus cam in a positive axis direction and from 1 back to 0 when it traverses the plus cam.
- The minus cam signal changes level in response to every positive edge of the plus cam signal.

Note

The described response of the plus cam applies on **condition**that:

plus cam - minus cam < 180 degrees

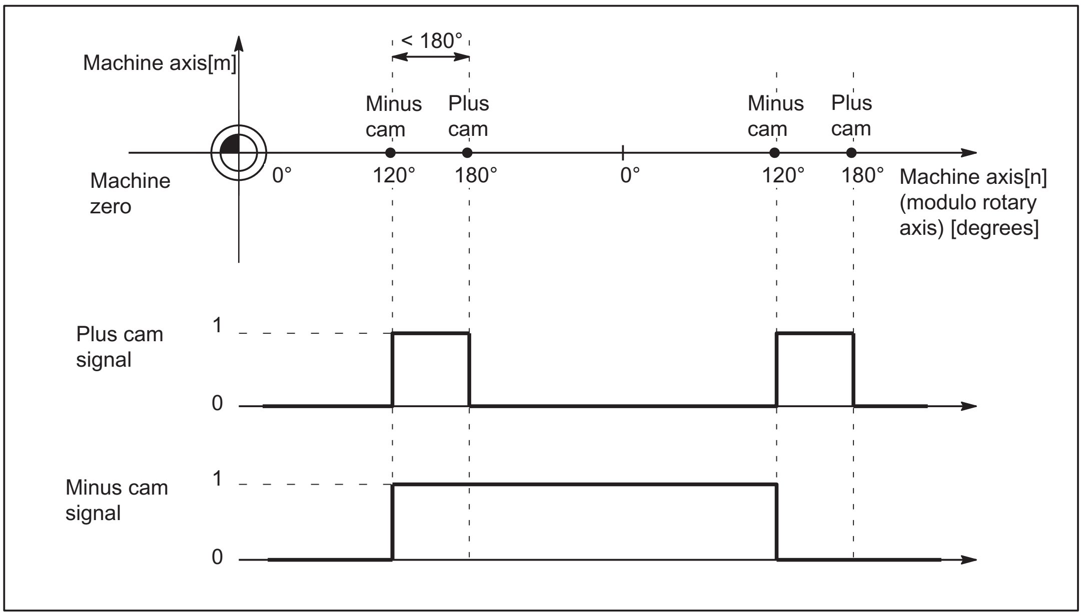


Figure 8-3 Software cams for modulo rotary axis (plus cam - minus cam < 180 degrees)

The signal change of the minus cam makes it possible to detect traversal of the cam even if the cam range is set so small that the PLC cannot detect it reliably.

Both cam signals can be output to the PLC and to the NCK I/Os. Separate output of the plus and minus cam signals makes it easy to detect whether the axis is within or outside the plus or minus cam range.

If this condition (plus cam - minus cam < 180 degrees) is not fulfilled or if the minus cam is set to a greater value than the plus cam, then the response of the plus cam signal is inverted. The response of the minus cam signal remains unchanged.

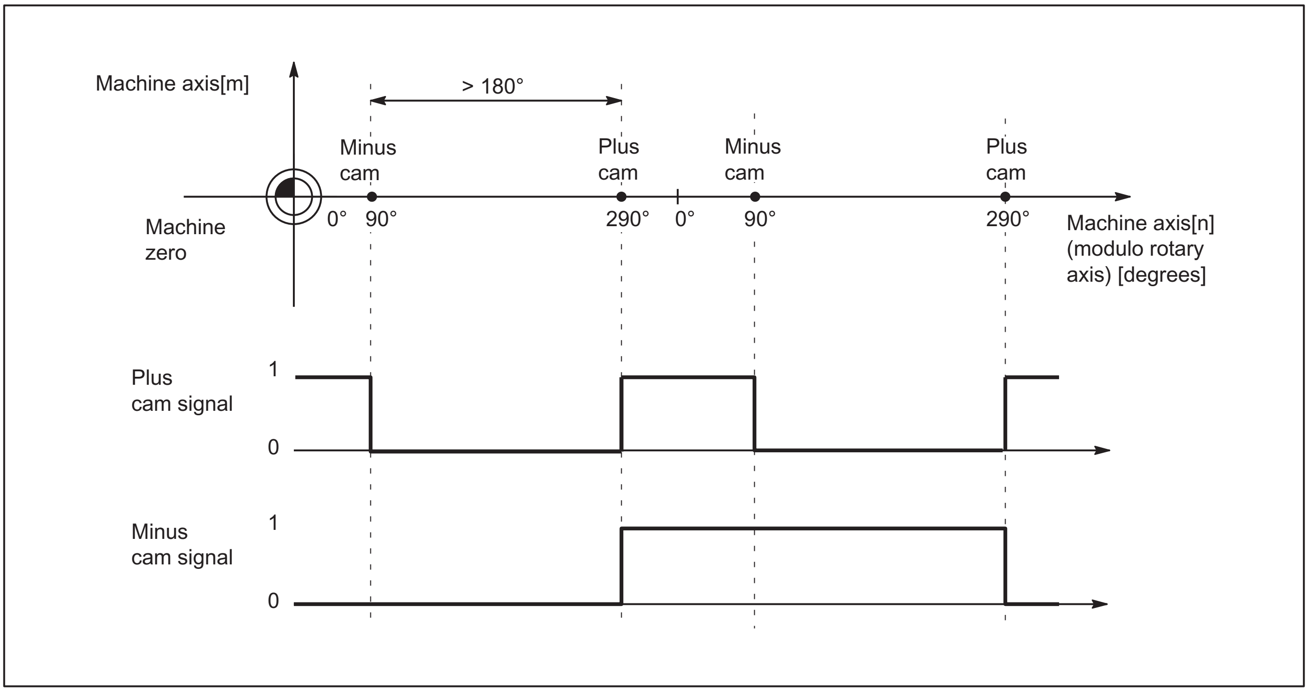


Figure 8-4 Software cams for modulo rotary axis (plus cam - minus cam > 180 degrees)

8.2.2 Generation of cam signals with gated output

The plus and minus cam output signals are gated in the case of:

- timer-controlled cam signal output to the four onboard outputs on the NCU
- Output to the NCK I/O, if the 2nd byte in the following machine data was not specified (= "0"):

  MD10470 SW_CAM_ASSIGN_FASTOUT_2

  ...

  MD10473 SW_CAM_ASSIGN_FASTOUT_4

Linear axes

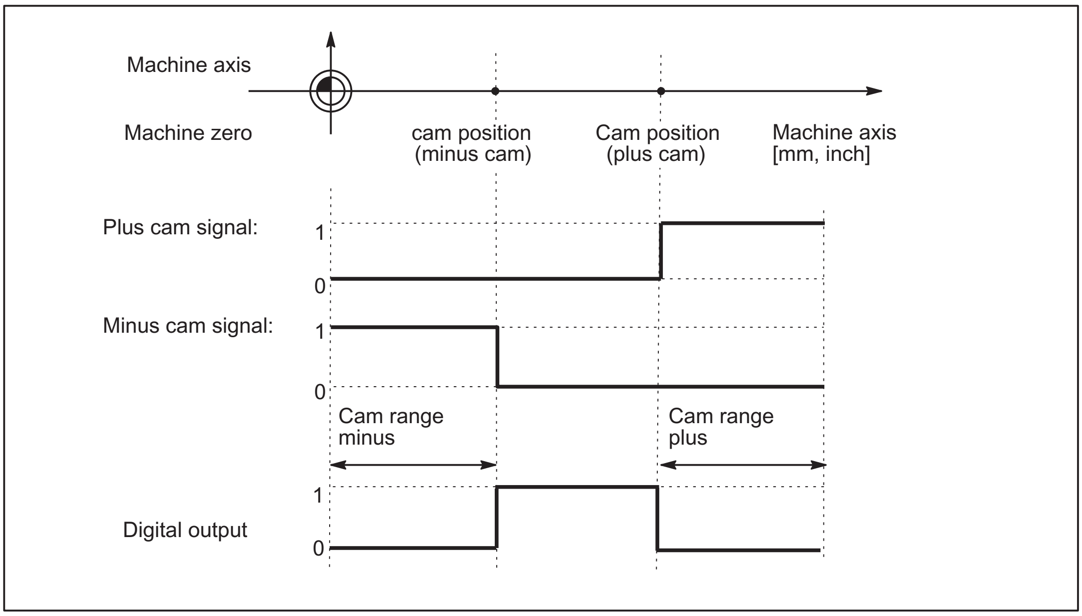


Figure 8-5 Position switching signals for linear axis (minus cam < plus cam)

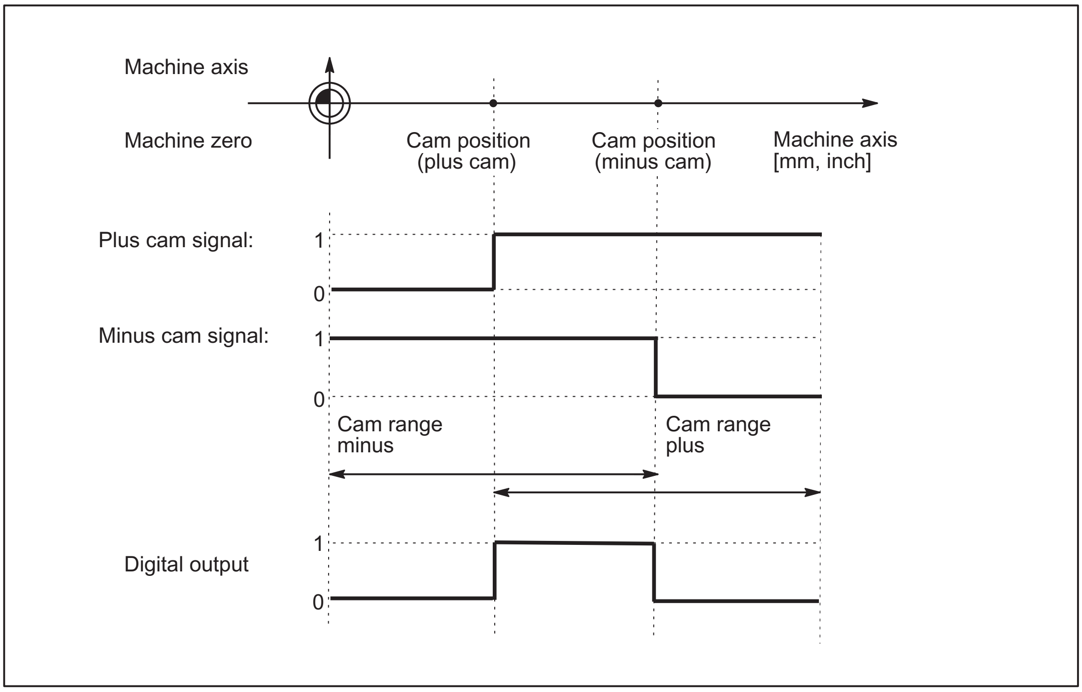


Figure 8-6 Position switching signals for linear axis (plus cam < minus cam)

Modulo rotary axis

The default signal response for modulo rotary axes is dependent on the cam width:

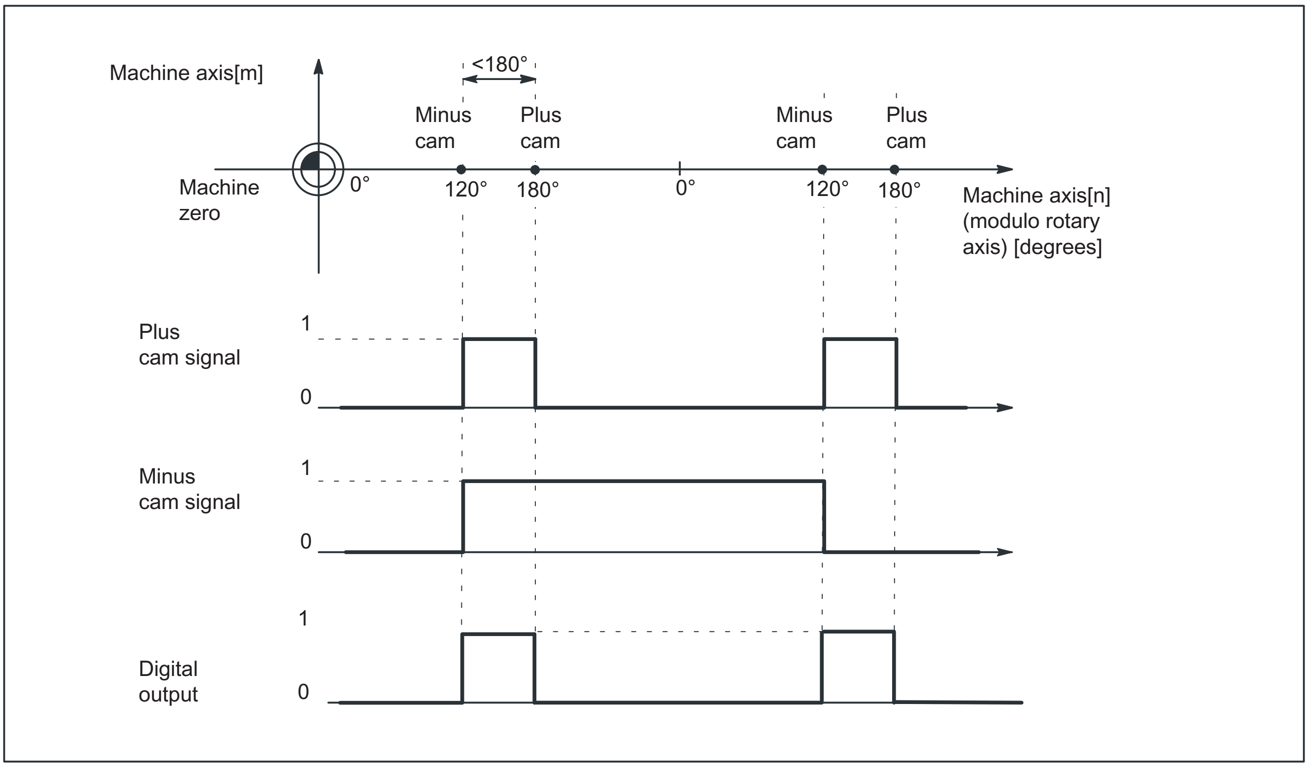


Figure 8-7 Software cams for modulo rotary axis (plus cam - minus cam < 180 degrees)

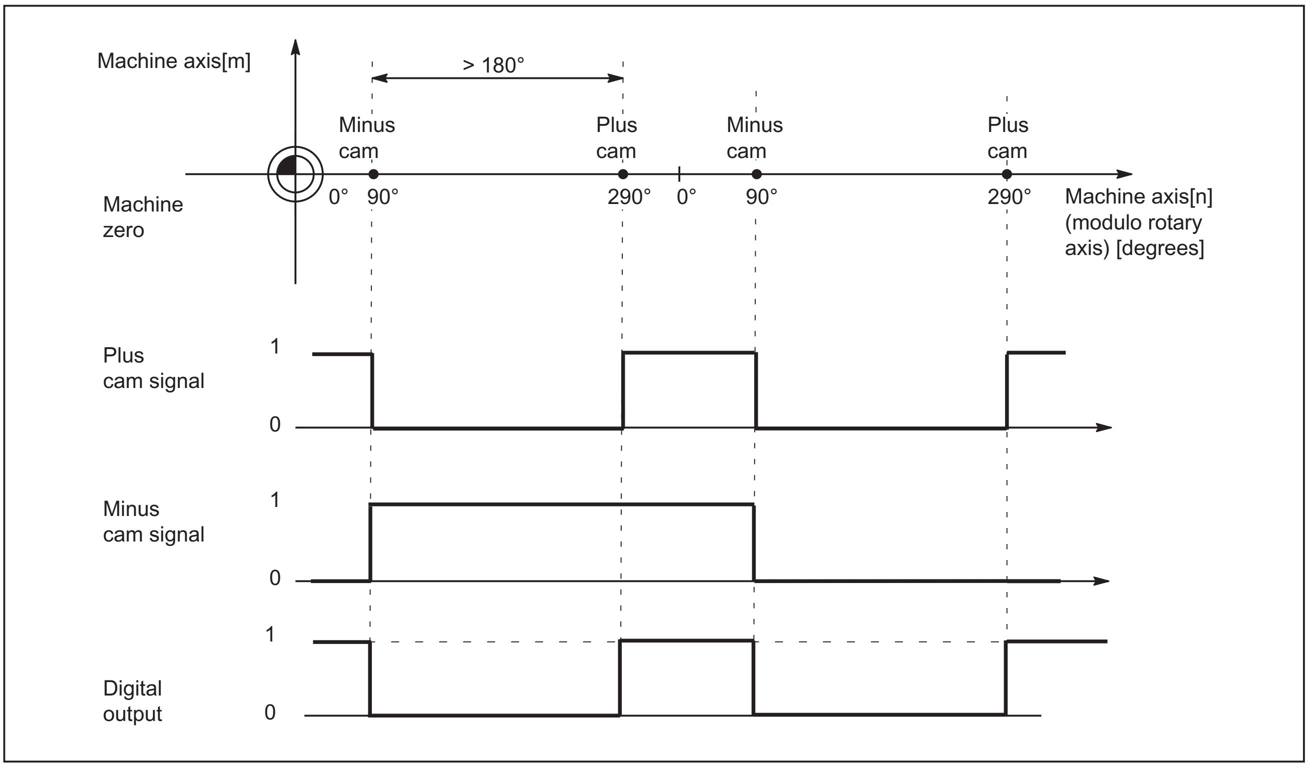


Figure 8-8 Software cams for modulo rotary axis (plus cam - minus cam > 180 degrees)

Suppression of signal inversion

With the following setting, selection of signal inversion for "plus cam - minus cam > 180 degrees" can be suppressed.

MD10485 SW_CAM_MODE bit 1=1

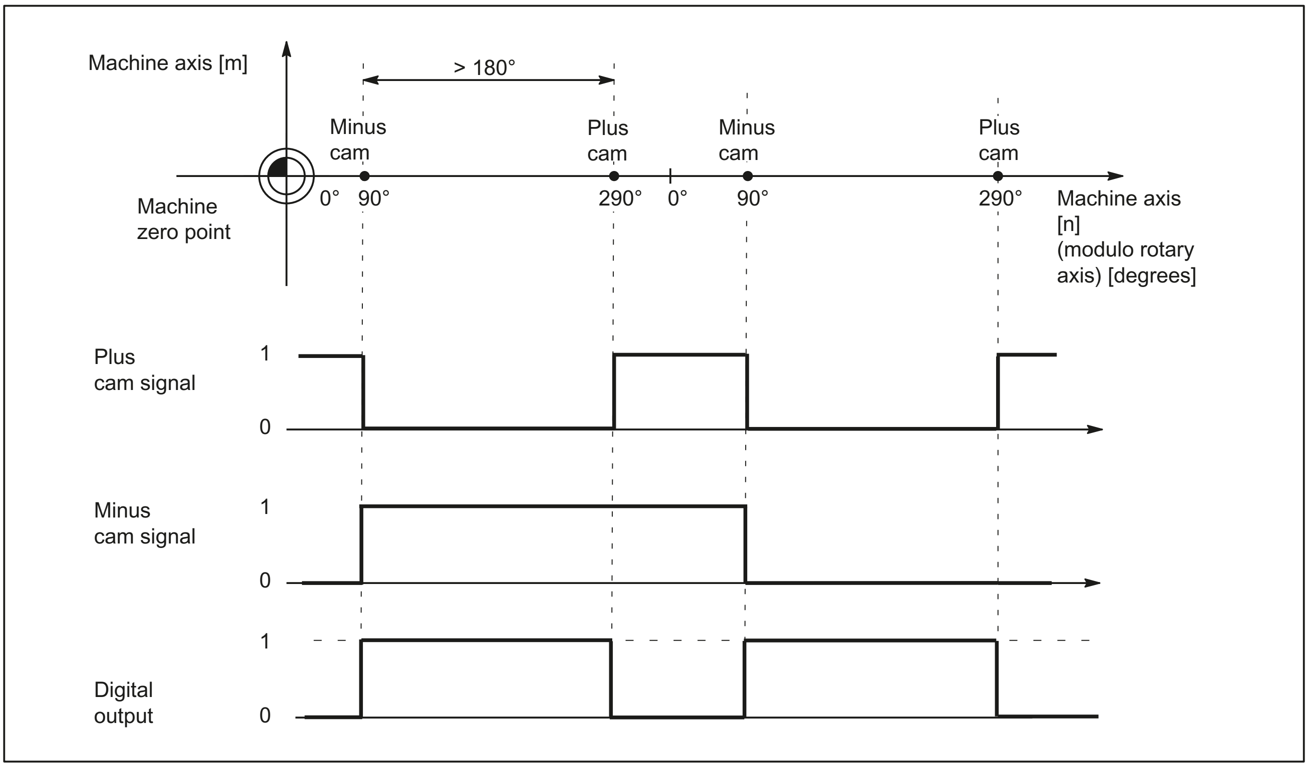


Figure 8-9 Software cams for modulo rotary axis (plus cam - minus cam > 180 degrees) and suppression of signal inversion

8.2.3 Cam positions

Setting cam positions

The positions of the plus and minus cams are defined using general setting data:

• SD41500 SW_CAM_MINUS_POS_TAB_1[n]	Position of minus cams 1 - 8
• SD41501 SW_CAM_PLUS_POS_TAB_1[n]	Position of plus cams 1 – 8
• SD41502 SW_CAM_MINUS_POS_TAB_2[n]	Position of minus cams 9 - 16
• SD41503 SW_CAM_PLUS_POS_TAB_2[n]	Position of plus cams 9 - 16
• SD41504 SW_CAM_MINUS_POS_TAB_3[n]	Position of minus cams 17 - 24

- SD41505 SW_CAM_PLUS_POS_TAB_3[n] Position of plus cams 17 - 24
- SD41506 SW_CAM_MINUS_POS_TAB_4[n] Position of minus cams 25 - 32
- SD41507 SW_CAM_PLUS_POS_TAB_4[n] Position of plus cams 25 - 32

Note

Owing to the grouping of cam pairs (eight in each group), it is possible to assign different access authorization levels (e.g. for machine-related and workpiece-related cam positions). The positions are entered in the machine coordinate system. No check is made with respect to the maximum traversing range.

Dimension system metric/inch

With the setting:
MD10260 CONVERT_SCALING_SYSTEM = 1
, the cam positions no longer refer to the configured basic dimension system, but to the dimension system set in the following machine data:

MD10270 POS_TAB_SCALING_SYSTEM (measuring system of position tables)

Value	Meaning
0	Metric
1	inch

MD10270 therefore defines the dimension system for position data from setting data SD41500 … SD41507.

A switchover with `G70`/`G71` or `G700`/`G710` has no effect.

Sensing of cam positions

To set the cam signals, the actual position of the axes is compared to the cam position.

Writing/reading of cam positions

The setting data can be read and written via HMI, PLC and part program.

Accesses from the part program are not synchronous to machining.
Synchronization can only be achieved by means of a programmed block preprocessing stop (`STOPRE` command).

It is possible to read and write the cam positions with FB2 and FB3 in the PLC user program.

Axis/cam assignment

A cam pair is assigned to a machine axis using the general machine data:

MD10450 SW_CAM_ASSIGN_TAB[n] (assignment of software cams to machine axes)

Note

Changes to an axis assignment take effect after the next NCK power-up.

Cam pairs to which no axis is assigned are not active.

A cam pair can only be assigned to one machine axis at a time.

Several cam pairs can be defined for one machine axis.

8.2.4 Lead/delay times (dynamic cam)

Function

To compensate for any delays, it is possible to assign two lead or delay times with additive action to each minus and plus cam for the cam signal output.

The two lead or delay times are entered in a machine data and a setting data.

Note

The input of negative time values causes a delay in the output of cam signals.

Lead or delay time in machine data

The **first** lead or delay time is entered in the following general machine data:

• MD10460 SW_CAM_MINUS_LEAD_TIME[n]	(lead or delay time at the minus cams)
• MD10461 SW_CAM_PLUS_LEAD_TIME[n]	(lead or delay time at the plus cams)

For example, the following can be entered into the machine data:

- Constant internal delay times between actual value sensing and cam signal output (e.g. as determined by an oscilloscope)
- Constant external delay times

Lead or delay time in setting data

The **second** lead or delay time is entered into the following general setting data:

- SD41520 SW_CAM_MINUS_TIME_TAB_1[n] (lead or delay time at the minus cams 1 – 8)
- SD41521 SW_CAM_PLUS_TIME_TAB_1[n] (lead or delay time at the plus cams 1 – 8)
- SD41522 SW_CAM_MINUS_TIME_TAB_2[n] (lead or delay time at the minus cams 9 – 16)
- SD41523 SW_CAM_PLUS_TIME_TAB_2[n] (lead or delay time at the plus cams 9 – 16)
- SD41524 SW_CAM_MINUS_TIME_TAB_3[n] (lead or delay time at the minus cams 17 – 24)
- SD41525 SW_CAM_PLUS_TIME_TAB_3[n] Lead or delay time on plus cams 17 - 24
- SD41526 SW_CAM_MINUS_TIME_TAB_4[n] (lead or delay time at the minus cams 25 – 32)
- SD41527 SW_CAM_PLUS_TIME_TAB_4[n] (lead or delay time at the plus cams 25 – 32)

Delay times which may change during machining must, for example, be entered in the setting data.

8.3 Output of cam signals

8.3.1 Activating

The status of the cam (cam signals) can be output to the PLC as well as to the NCK I/Os.

Activation of cam signal output

The output of cam signals of an axis is activated via the NC/PLC interface signal:

DB31, ... DBX2.0 (cam activation)

Check-back signal to PLC

The successful activation of all cams of an axis is signaled back to the PLC using the following NC/PLC interface signal:

DB31, ... DBX62.0 (cams active)

Note

The activation can be linked with other conditions by the PLC user (e.g. axis referenced, reset active).

8.3.2 Output of cam signals to PLC

Output to PLC

The status of the cam signals for all machine axes with activated software cams is output to the PLC.

The status is output in the IPO cycle and is transferred to the PLC asynchronously.

Minus cam signals

The status of the minus cam signals is entered into the following NC/PLC interface signals:

DB10 DBX110.0 to 113.7 (minus cam signal 1 to 32)

Plus cam signals

The status of the plus cam signals is entered into the following NC/PLC interface signals:

DB10 DBX114.0 to 117.7 (plus cam signals 1 to 32)

Note

If no measuring system has been selected or NC/PLC interface signal DB31, ... DBX2.0 (cam activation) is set to 0, then the following NC/PLC interface signals are also set to 0:

- DB10 DBX110.0-113.7 (minus cam signals 1-32)
- DB10 DBX114.0-117.7 (plus cam signals 1-32)
- DB31, ... DBX62.0 (cams active)

8.3.3 Output of cam signals to NCK I/Os in position control cycle

Signal output in position control cycle

For cams assigned to an HW byte via machine data MD10470 to MD10473 (see Section "Hardware assignment"), the signal is output in the position control cycle.

The four onboard outputs on the NCU and a total of 32 optional external NCK outputs are available as the digital outputs of the NCK I/Os (see Section "A4: Digital and analog NCK I/Os for SINUMERIK 840D sl (Page 25)").

Hardware assignment

The assignment to the hardware bytes used is made for each eight cam pairs in the following general machine data:

• MD10470 SW_CAM_ASSIGN_FASTOUT_1	Hardware assignment for output of cams 1 - 8 to NCK I/Os
• MD10471 SW_CAM_ASSIGN_FASTOUT_2	Hardware assignment for output of cams 9 - 16 to NCK I/Os
• MD10472 SW_CAM_ASSIGN_FASTOUT_3	Hardware assignment for output of cams 17 - 24 to NCK I/Os
• MD10473 SW_CAM_ASSIGN_FASTOUT_4	Hardware assignment for output of cams 25 - 32 to NCK I/Os

---

Note

It is possible to define one HW byte for the output of eight minus cam signals and one HW byte for the output of eight plus cam signals in each machine data.

In addition, the output of the cam signals can be inverted with the two machine data.

If the 2nd byte is not specified (= "0"), then the 8 cams are output as a logic operation of the minus and plus cam signals via the 1st HW byte using the 1st inversion screen form.

Status query in the part program

The status of the HW outputs can be read in the part program with main run variable **$A_OUT[n]** (n = no. of output bit).

Switching accuracy

Signals are output to the NCK I/Os or onboard outputs in the position control cycle.
As a result of the time scale of the position control cycle, the switching accuracy of the cam signals is limited as a function of the velocity.

The following applies:

Delta pos = V_{act} * position control cycle

with:	Delta pos =	switching accuracy (depending on position control cycle)
	V_{act} =	Current axis velocity

Example:

V_{act} = 20 m/min, PC cycle = 4 ms	Delta pos = 1.33 mm
V_{act} = 2000 rpm, PC cycle = 2 ms	Delta pos = 24 degrees

8.3.4 Timer-controlled cam signal output

Timer-controlled output

A significantly higher degree of accuracy can be achieved by outputting the cam signals independently of the position control cycle using a timer interrupt.

The following machine data can be used to select the timer-controlled output to the 4 NCU onboard outputs for 4 cam pairs:

MD10480 SW_CAM_TIMER_FASTOUT_MASK (screen form for the output of cam signals via timer interrupts on the NCU)

In this case, the minus and plus signals of a cam pair are logically combined for output as one signal.

Signal generation

Previously, it had to be specified in which way the signals to be logically combined should be generated. This is realized using bit 1 in machine data:

MD10485 SW_CAM_MODE (behavior of the software cams)

Bit	Value	Signal generation
1	0	Inversion of the signal behavior of the plus cam when: plus cam - minus cam ≥ 180 degrees
	1	**No** inversion of the signal behavior of the plus cam when: plus cam - minus cam ≥ 180 degrees

---

Note

This function operates independently of the HW assignment selected in MD10470 ... MD10473.

The onboard byte may not be used a multiple number of times.

Restrictions

The following applies to the mutual position of the cam positions:

Only **one** timer-controlled output takes place per interpolation cycle.

If signal changes for more than one cam pair are active in the same interpolation cycle, the output is prioritized:

The cam pair with the lowest number (1...32) determines the output time for **all** active signals, i.e. the signal change of the other cam pairs takes place at the same time.

PLC interface

The NCK image of the onboard outputs and the status of the plus and minus cams is displayed on the PLC interface.

However, these signals are irrelevant or correspondingly inaccurate for the **timer-controlled** cam output version, as described in the following paragraphs. The signals for the plus and minus cams are generated (once) in synchronism with the interpolation cycle and transmitted together to the PLC.

Pulses shorter than an interpolation cycle are thus imperceptible on the PLC.
The onboard outputs are set and reset asynchronously to the interpolation cycle for each interrupt. The status of the onboard outputs is detected and transmitted to the PLC in synchronism with the update time of the PLC interface.

Depending on the current status at the moment the PLC interface is updated, pulses shorter than one interpolation cycle are not visible or are displayed stretched by one or several IPO cycles.

Further settings

The following bit must be set to "0" if the behavior described here is to be activated:

MD10485 SW_CAM_MODE bit 0 = 0

8.3.5 Independent, timer-controlled output of cam signals

Independent, timer-controlled cam output

Each switching edge is output separately per interrupt due to the timer-controlled, independent (of interpolation cycle) cam output.

The mutual influence of the cam signals is no longer applicable as a result of:

- single output per interpolation cycle
- output time determined by highest priority cam pair (lowest cam pair number)

A total of 8 timer-controlled cam outputs per interpolation cycle can be configured for setting/resetting the four onboard outputs. The signal states of the plus and minus cams are also made available as standard on the PLC interface for the timer-controlled variant, However, these signals are not relevant or are correspondingly inaccurate with a timer-controlled output.

Signal generation

Previously, it had to be specified in which way the signals to be logically combined should be generated. This is realized using bit 1 in machine data:

MD10485 SW_CAM_MODE (behavior of the software cams)

Bit	Value	Signal generation
1	0	Inversion of the signal behavior of the plus cam when: plus cam - minus cam ≥ 180 degrees
	1	**No** inversion of the signal behavior of the plus cam when: plus cam - minus cam ≥ 180 degrees

Settings

Cam pairs are assigned to onboard outputs using machine data:

MD10480 SW_CAM_TIMER_FASTOUT_MASK (screen form for the output of cam signals via timer interrupts on the NCU)

In addition, this type of processing must be explicitly selected:

MD10485 SW_CAM_MODE bit 0 = 1

Note

This function operates independently of the HW assignment selected in MD10470 … MD10473.

The onboard byte may not be used a multiple number of times.

8.4 Position-time cams

Position-time cams

The term "position-time cam" refers to a pair of software cams that can supply a pulse of a certain duration at a defined axis position.

Solution

The position is defined by a pair of software cams.
The pulse duration is defined by the lead/delay time of the plus cam.

Machine data can be used to specify that cam pairs with "minus cam position = plus cam position" should be processed as position-time cams.

Properties of position-time cams

- The pulse duration is independent of the axis velocity and travel direction reversal.
- The pulse duration is independent of changes in the axis position (Preset).
- Activation (rising signal edge) takes place only when the cam position is crossed.

  Moving the axis position (e.g. preset) does not trigger activation.
- A lead/delay time is operative for the minus cam and causes a time displacement of the pulse.
- Activation (ON edge) and pulse duration are independent of the travel direction.
- The cam is not deactivated if the cam position is crossed again when the cam is active (direction reversal).
- The cam time (pulse width) is not interrupted and the cam time not restarted when the cam position is crossed again.

  This behavior is particularly relevant with respect to modulo axes, i.e. if the cam time is greater than the modulo range crossing time, the cam is not switched in every revolution.

Settings

The following settings must be made to program a position-time cam:

- Position

  The position must be defined by a cam pair with which the minus cam position is equal to the plus cam position.

  This is defined using setting data:

  SD41500 SW_CAM_MINUS_POS_TAB_1
  ...
  SD41507 SW_CAM_PLUS_POS_TAB_4.
- Pulse duration

  The pulse duration is calculated by adding together the associated entries for the cam pair in:

  MD10461 SW_CAM_PLUS_LEAD_TIME[n]

  SD41521 SW_CAM_PLUS_TIME_TAB_1[n]...

  SD41527 SW_CAM_PLUS_TIME_TAB_4[n]
- Offset

  The time displacement of the position-time cam is calculated by adding together the associated entries for the cam pair in:

  MD10460 SW_CAM_MINUS_LEAD_TIME[n]

  SD41520 SW_CAM_MINUS_TIME_TAB_1[n]...

  SD41526 SW_CAM_MINUS_TIME_TAB_4[n]
- Mode

  MD10485 SW_CAM_MODE
  Bit 2 = 1 must be set in the machine data to ensure that all cam pairs with the same values for minus cam and plus cam positions are treated as position-time cams.

8.5 Supplementary Conditions

Availability of function "Software cams, position switching signals"

The function is an option ("position-switching signals/cam controller"), which must be assigned to the hardware through the license management.

8.6 Data lists

8.6.1 Machine data

8.6.1.1 General machine data

Number	Identifier: $MN_	Description
10260	CONVERT_SCALING_SYSTEM	Basic system switchover active
10270	POS_TAB_SCALING_SYSTEM	System of measurement of position tables
10450	SW_CAM_ASSIGN_TAB[n]	Assignment of software cams to machine axes
10460	SW_CAM_MINUS_LEAD_TIME[n]	Lead or delay time on minus cams 1 -16
10461	SW_CAM_PLUS_LEAD_TIME[n]	Lead or delay time on plus cams 1 -16
10470	SW_CAM_ASSIGN_FASTOUT_1	Hardware assignment for output of cams 1 -8 to NCK I/Os
10471	SW_CAM_ASSIGN_FASTOUT_2	Hardware assignment for output of cams 9 -16 to NCK I/Os
10472	SW_CAM_ASSIGN_FASTOUT_3	Hardware assignment for output of cams 17 -24 to NCK I/Os
10473	SW_CAM_ASSIGN_FASTOUT_4	Hardware assignment for output of cams 25 -32 to NCK I/Os
10480	SW_CAM_TIMER_FASTOUT_MASK	Screen form for output of cam signals via timer interrupts to NCU
10485	SW_CAM_MODE	Response of SW cams

8.6.2 Setting data

8.6.2.1 General setting data

Number	Identifier: $SN_	Description
41500	SW_CAM_MINUS_POS_TAB_1[n]	Position of minus cams 1 -8
41501	SW_CAM_PLUS_POS_TAB_1[n]	Position of plus cams 1 -8
41502	SW_CAM_MINUS_POS_TAB_2[n]	Position of minus cams 9 -16
41503	SW_CAM_PLUS_POS_TAB_2[n]	Position of plus cams 9 -16
41504	SW_CAM_MINUS_POS_TAB_3[n]	Position of minus cams 17 -24
41505	SW_CAM_PLUS_POS_TAB_3[n]	Position of plus cams 17 -24
41506	SW_CAM_MINUS_POS_TAB_4[n]	Position of minus cams 25 -32
41507	SW_CAM_PLUS_POS_TAB_4[n]	Position of plus cams 25 -32
41520	SW_CAM_MINUS_TIME_TAB_1[n]	Lead or delay time on minus cams 1 -8
41521	SW_CAM_PLUS_TIME_TAB_1[n]	Lead or delay time on plus cams 1 -8
41522	SW_CAM_MINUS_TIME_TAB_2[n]	Lead or delay time on minus cams 9 -16
41523	SW_CAM_PLUS_TIME_TAB_2[n]	Lead or delay time on plus cams 9 -16
41524	SW_CAM_MINUS_TIME_TAB_3[n]	Lead or delay time on minus cams 17 -24
41525	SW_CAM_PLUS_TIME_TAB_3[n]	Lead or delay time on plus cams 17 -24
41526	SW_CAM_MINUS_TIME_TAB_4[n]	Lead or delay time on minus cams 25 -32
41527	SW_CAM_PLUS_TIME_TAB_4[n]	Lead or delay time on plus cams 25 -32

8.6.3 Signals

8.6.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Cam activation	DB31, ...DBX2.0	-

8.6.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Cams active	DB31, ...DBX62.0	-

9 N4: Own channel - only 840D sl

9.1 Brief Description

Subfunctions

The functions specific to punching and nibbling operations comprise the following:

- Stroke control
- Automatic path segmentation
- Rotatable punch and die
- Clamp protection

They are activated and deactivated via language commands.

9.2 Stroke control

9.2.1 General information

Functionality

The stroke control is used in the actual machining of the workpiece. The punch is activated via an NC output signal when the position is reached. The punching unit acknowledges its punching motion with an input signal to the NC. No axis may move within this time period. Repositioning takes place after the punching operation.

High-speed signals

"High-speed signals" are used for direct communication between the NC and punching unit. Combined with the punch, they allow a large number of holes to be punched per minute since the punch positioning times are interpreted as machining delays.

PLC signals

PLC interface signals are used for non-time-critical functions such as enabling and monitoring.

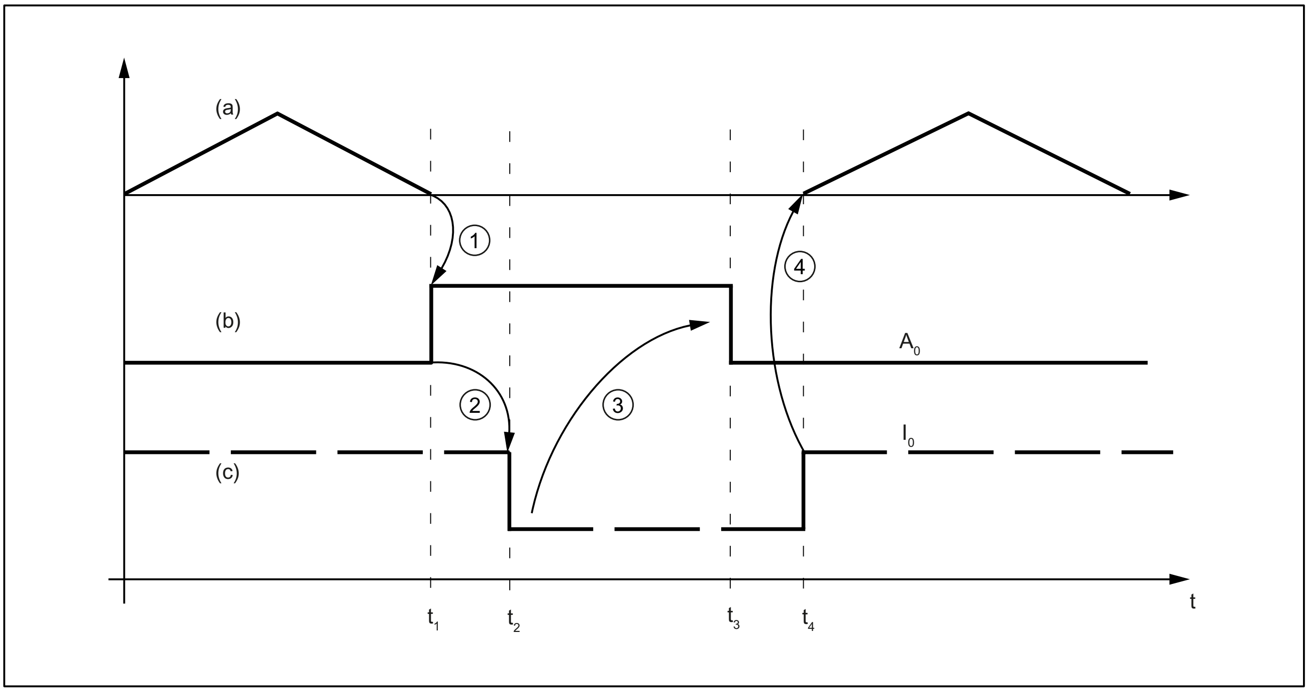
9.2.2 High-speed signals

Functionality

High-speed signals are used to synchronize the NC and punching unit. On the one hand, they are applied via a high-speed output to ensure that the punch stroke is not initiated until the metal sheet is stationary. On the other, they are applied via a high-speed input to ensure that the sheet remains stationary while the punch is active.

The high-speed digital inputs and outputs on the control are used to drive the punching unit.

The following signal chart illustrates the signal sequence.



(a) Axis motion of the machine as function v(t)

(b) "Stroke initiation" signal

(c) "Stroke active" signal

Figure 9-1 Signal chart

Note

The "Stroke active" signal is high-active for reasons relating to open-circuit monitoring.

The chronological sequence of events for punching and nibbling is controlled by the two signals A_0 and E_0:

A_0	Set by the NCK and identical to stroke initiation.
E_0	Defines the status of the punching unit and identical to the "Stroke active" signal.

The signal states characterize and define times t_1 to t_4 in the following way:

t_1	The motion of the workpiece (metal sheet) in relation to the punch is completed at instant t_1. Depending on the criterion defined for stroke initiation (see Section "Criteria for stroke initiation (Page 558)"), the high-speed output A_0 is set for punch initiation ①.
t_2	The punching unit signals a punch movement via high-speed input E_0 at instant t_2. This is triggered by signal A_0 ②. For safety reasons, signal E_0 is high-active (in the case of an open circuit, "Stroke active" is always set and the axes do not move). The "Stroke active" signal is not reset again until the tool has moved away from the metal sheet (t_4).
t_3	The NC reacts to the "Stroke active" signal at instant t_3 by canceling the "Stroke initiation" signal ③. From this point in time onwards, the NC is in a waiting state. It simply waits for cancellation of the "Stroke active" signal so it can initiate the next axis motion. The next stroke can be initiated only after signal A_0 has disappeared.
t_4	The punching operation is complete at instant t_4, i.e. the punch has exited from the metal sheet again. The NC reacts to a signal transition in signal E_0 by starting an axis motion ④. The reaction of the NC to a signal edge change ④ is described in the section headed "Axis start after punching".

Note

The stroke time is determined by the period $\Delta t_h = t_4 - t_1$.

Reaction times at instant t_4 between the signal transition of E_0 and the start of the axis motion must also be added.

9.2.3 Criteria for stroke initiation

Initiate a stroke

The stroke initiation must be set, at the earliest, for the point in time at which it can be guaranteed that the axes have reached a standstill. This ensures that at the instant of punching, there is absolutely no relative movement between the punch and the metal sheet in the machining plane.

The following diagram shows the various criteria that can be applied to stroke initiation.

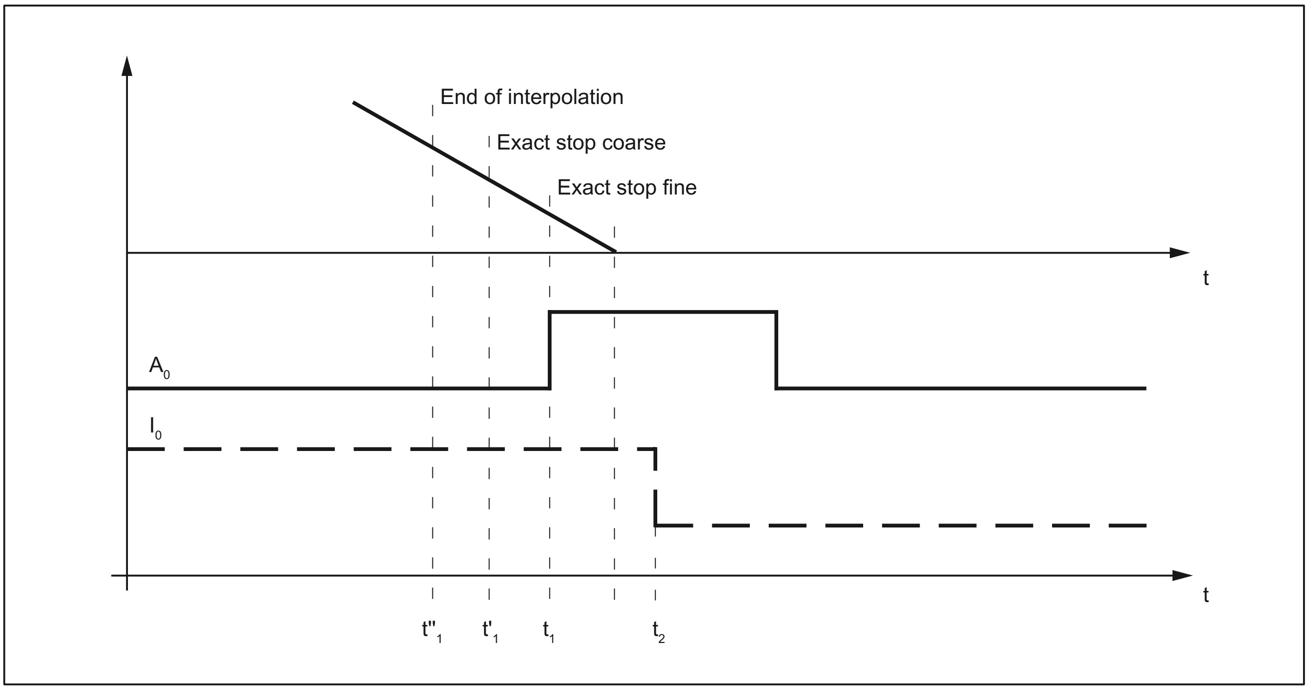


Figure 9-2 Signal chart: Criteria for stroke initiation

The time interval between t_1 and t_2 is determined by the reaction of the punching unit to setting of output A_0. This cannot be altered, but can be utilized as a lead time for minimizing dead times.

The diagram above shows the default setting with which the output is set when the "Exact stop fine window" is reached (`G601`; default setting of G group 12). The punch initiation times t''_1 and t'_1 are programmed by means of `G602` and `G603` (see table below).

Programming	Activation	Description
`G603`	Stop interpolation	The interpolation reaches the block end. In this case, the axes continue to move until the overtravel has been traversed, i.e. the signal is output at an appreciable interval before the axes have reached zero speed (see t''_1).

Programming	Activation	Description
`G602`	Reach the coarse in-position window	The signal is output once the axes have reached the coarse in-position window. If this criterion is selected for stroke initiation output, then the instant of stroke initiation can be varied through the size of interpolation window (see t'_1).
`G601`	Reach the fine in-position window	In this case, it can always be ensured that the machine will have reached a standstill at the instant of punching provided that the axis data are set appropriately. However, this variant also results in a maximum dead time (see t_1).

Note

The initial setting of the G group with `G601`, `G602` and `G603` (G group 12) is defined via machine data:

MD20150 $MC_GCODE_RESET_VALUES[11]

The default setting is `G601`.

G603

Depending on velocity and machine dynamics, approximately 3 - 5 interpolation cycles are processed at the end of interpolation before the axes reach zero speed.

MD26018 $MC_NIBBLE_PRE_START_TIME
In conjunction with the above machine data, it is possible to delay, and therefore optimize, the instant between reaching the end of interpolation and setting the high-speed output for "Stroke ON".

The following setting data is available in addition to MD26018:

SD42402 $SC_NIBPUNCH_PRE_START_TIME

SD42402 can be changed from the part program and therefore adapted to the punching process depending on the progress of the part program run.

The following applies to the delay time:

MD26018 = 0	→ SD42402 is operative
MD26018 ≠ 0	→ MD26018 is operative

If the "Punching with dwell time, `PDELAYON`" is active, then the dwell time programmed in connection with this function is active. Neither MD26018 nor SD42402 is operative in this case.

9.2.4 Axis start after punching

Input signal "Stroke ON"

The start of an axis motion after stroke initiation is controlled via input signal "Stroke ON".

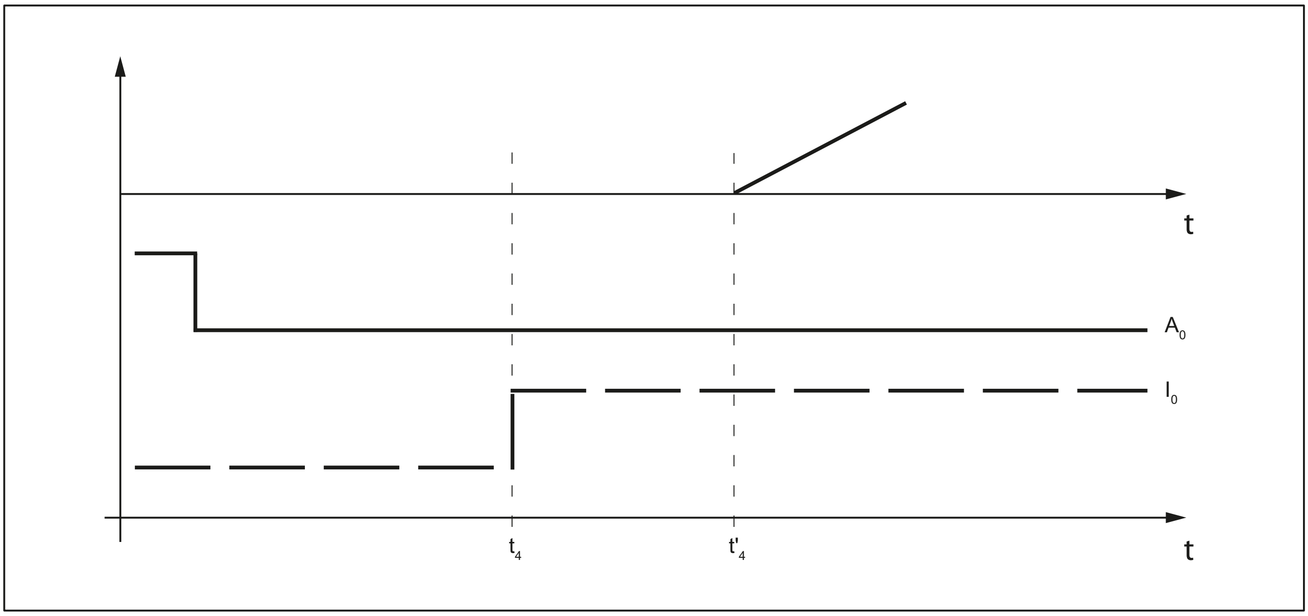


Figure 9-3 Signal chart: Axis start after punching

In this case, the time interval between t_4 and t'_4 acts as a switching-time-dependent reaction time. It is determined by the interpolation sampling time and the programmed punching/nibbling mode.

PON/SON

When the punching unit is controlled via PON/SON, the maximum delay time is calculated to be:

$|t'_4 - t_4| = 3$ x interpolation cycles

PONS/SONS

If the punch is controlled by PONS/SONS, the delay time is determined by:

$|t'_4 - t_4| \leq 3$ x position controller cycles

Prerequisites: Stroke time ($t_4 - t_2$) > 4 interpolation cycles

9.2.5 PLC signals specific to punching and nibbling

Function

In addition to the signals used for direct stroke control, channel-specific PLC interface signals are also available. These are used both to control the punching process and to display operational states.

Signals

Signal	Activation
DB21, ... DBX3.0 (no stroke enable)	Prevents the NC from initiating a punching operation. The NC waits until the enable signal is available before continuing the part program.
DB21, ... DBX3.2 (stroke suppression)	Allows the part program to be processed without initiating a punching operation (dry run). With active path segmentation, the axes traverse in "Stop and go" mode.
DB21, ... DBX3.4 (delayed stroke)	Activates delayed stroke output if permitted by `PDELAYON`.
DB21, ... DBX3.1 (manual stroke suppression)	Enables the operator to initiate a punching operation (controlled via the PLC) without executing the part program. Manual stroke initiation is acknowledged with signal: DB21, .. DBX38.1 (acknowledgement of manual stroke initiation)

9.2.6 Punching and nibbling-specific reactions to standard PLC signals

DB21, ... DBX12.3 (feed stop)

With interface signal:
DB21, ... DBX12.3 (feed stop)
, the NC reacts in the following way with respect to the stroke control:

Signal is detected in advance of instant t_1:	Stroke initiation is suppressed. The next stroke is not initiated until the next start or until the "Feed stop" signal has been canceled. Machining is then continued as if there had been no interruption.
Signal is detected at instant t_1:	The current stroke is executed to completion. The NC then dwells in the state characterized by t_4. To allow it to respond in this manner, time monitoring of the "Stroke active" and "Stroke initiation" signals is dispensed with.

9.2.7 Signal monitoring

Oscillating signal

Owing to aging of the punch hydraulics, overshooting of the punch may cause the "Stroke active" signal to oscillate at the end of a stroke.

In this case, an alarm (22054 "undefined punching signal") can be generated as a function of machine data:
MD26020 $MC_NIBBLE_SIGNAL_CHECK
.

Reset response

In the case of an NCK reset, the interface signal:
DB21, ... DBX38.0 (stroke initiation active)
is canceled immediately without acknowledgement by the high-speed input.

A currently activated stroke cannot be suppressed.

9.3 Activation and deactivation

9.3.1 Language commands

Punching and nibbling functions are activated and deactivated via configurable language commands. These replace the special M functions that were used in earlier systems.

References:
Programming Manual, Production Planning

Groups

The language commands are subdivided into the following groups:

Group 35	
The actual punching and nibbling-specific functions are activated and deactivated by means of the following language commands:	
`PON`	= punching ON
`SON`	= nibbling ON
`PONS`	= punching ON, activated in the position controller
`SONS`	= nibbling ON, activated in the position controller
`SPOF`	= punching/nibbling OFF

Group 36	
This group includes the commands which have only a preparatory character and which determine the real nature of the punching function:	
PDELAYON	= punching with delay ON
PDELAYOF	= punching with delay OFF
Since the PLC normally needs to perform some preliminary tasks with respect to these preparatory functions, they are programmed before the activating commands.	

Group 38	
This group contains the commands for switching over to a second punch interface. It can be used, for example, for a second punching unit or set of hammer shears. A second I/O pair which can be used for punching functionality is defined via machine data.	
SPIF1	= first interface is active
SPIF2	= second interface is active

Note

Only one function at a time can be active within a G code group (similar, for example, to the various interpolation modes G0, G1, G2, G3, etc. which are also mutually exclusive).

SPOF

Punching and nibbling OFF

The SPOF function terminates all punching and nibbling functions. In this state, the NCK responds neither to the "Stroke active" signal nor to the PLC signals specific to punching and nibbling functions.

If SPOF is programmed together with a travel command in one block (and in all further blocks if punching/nibbling is not activated with SON or PON), the machine approaches the programmed position without the initiation of a punching operation. SPOF deselects SON, SONS, PON and PONS and corresponds to the Reset condition.

Programming example:

Program code	Comment
:	
:	
N20 G90 X100 SON	; activate nibbling
N25 X50 SPOF	; Deactivate nibbling,
	; position without stroke initiation
:	
:	

SON

Nibbling ON

`SON` activates the nibbling function and deselects the other functions in G group35 (e.g. `PON`).

In contrast to punching, the first stroke is made at the start point of the block with the activating command, i.e. before the first machine motion.

`SON` has a modal action, i.e. it remains active until either `SPOF` or `PON` is programmed or until the program end is reached.

The stroke initiation is suppressed in blocks without traversing information relating to the axes designated as punching or nibbling axes (typically those in the active plane). If a stroke still needs to be initiated, then one of the punching/nibbling axes must be programmed with a 0 traversing path. If the first block with `SON` is a block without traversing information of the type mentioned, then only one stroke takes place in this block since the start and end points are identical.

Programming example:

Program code	Comment
:	
:	
N70 X50 SPOF	; position without punch initiation
N80 X100 SON	; Activate nibbling, initiate a stroke before the
	; motion (X=50) and on completion of the programmed
	; movement (X=100)
:	
:	

SONS

Nibbling ON (in position control cycle)

`SONS` behaves in the same way as `SON`. The function is activated in the position control cycle, thus allowing time-optimized stroke initiation and an increase in the punching rate per minute.

PON

Punching ON

`PON` activates the punching function and deactivates `SON`.

`PON` has a modal action like `SON`.

In contrast to `SON`, however, a stroke is not executed until the end of the block or, in the case of automatic path segmentation, at the end of a path segment. PON has an identical action to `SON` in the case of blocks which contain no traversing information.

Programming example:

Program code	Comment
:	
:	
N100 Y30 SPOF	; position without punch initiation
N110 Y100 PON	; Activate punching, punch initiation at the end of the
	; positioning operation (Y=100)
:	
:	

PONS

Punching ON (in position control cycle)

`PONS` behaves in the same way as `PON`. For explanation, please refer to `SONS`.

PDELAYON

Punching with delay ON

`PDELAYON` is a preparatory function. This means that `PDELAYON` is generally programmed before `PON`. The punch stroke is output with a delay when the programmed end position is reached.

The delay time in seconds is programmed in setting data:

SD42400 $SC_PUNCH_DWELLTIME

If the defined value cannot be divided as an integer into the interpolation clock cycle, then it is rounded to the next divisible integer value.

The function has a modal action.

PDELAYOF

Punching with delay OFF

`PDELAYOF` deactivates punching with delay function, i.e. the punching process continues normally. `PDELAYON` and `PDELAYOF` form a G code group.

Programming example:

`SPIF2` activates the second punch interface, i.e. the stroke is controlled via the second pair of high-speed I/Os (see Section "Channelspecific machine data (Page 596)", MD26004 and MD26006).

Program code	Comment
:	
:	
N170 PDELAYON X100 SPOF	; Position without punch initiation, activate
	; the delayed punch initiation
:	
:	

Program code	Comment
N180 X800 PON	; activate punching. After reaching the
	; end position, the punch stroke is output delayed
:	
:	
N190 PDELAYOF X700	; Deactivate punching with delay, normal
	; initiation of a punching operation. End of the programmed
	; movement
:	
:	
:	

SPIF1

Activation of first punch interface

`SPIF1` activates the first punch interface, i.e. the stroke is controlled via the first pair of high-speed I/Os (see Section "Channelspecific machine data (Page 596)", MD26004 and MD26006).

The first punch interface is always active after a reset or control system power up. If only one interface is used, then it need not be programmed.

SPIF2

Activation of second punch interface

`SPIF2` activates the second punch interface, i.e. the stroke is controlled via the second pair of high-speed I/Os (see Section "Channelspecific machine data (Page 596)", MD26004 and MD26006).

Programming example:

Program code	Comment
:	
:	
N170 SPIF1 X100 PON	; At the end of the block, a stroke is initiated at
	; the first fast output. The signal "stroke
	; active" is monitored at the first input.
:	
:	
:	
:	
N180 X800 SPIF2	; The second stroke is initiated at the
	; second fast output. "Stroke active" signal
	; is monitored at the second input.
:	
:	

Program code	Comment
:	
N190 SPIF1 X700	; All further strokes are controlled
	; with the first interface.
:	

9.3.2 Functional expansions

Alternate interface

Machines that alternately use a second punching unit or a comparable medium can be switched over to a second I/O pair.

The second I/O pair can be defined via the following machine data:

MD26004 $MC_NIBBLE_PUNCH_OUTMASK

MD26006 $MC_NIBBLE_PUNCH_INMASK

The interface is switched by command `SPIF1` or `SPIF2`.

Full punching/nibbling functionality is available on both interfaces.

Example:

Hardware assignment for stroke control

Define the high-speed byte in each case on the CPU as a high-speed punch interface:

MD26000 $MC_PUNCHNIB_ASSIGN_FASTIN = 'H00030001' → Byte 1

MD26002 $MC_PUNCHNIB_ASSIGN_FASTOUT = 'H00000001'

Remark:

The first and second bits are inverted.

Screen form for high-speed input and output bits:

MD26004 $MC_NIBBLE_PUNCH_OUTMASK[0]	= 1	First interface output bit → Bit 1 SPIF1
MD26004 $MC_NIBBLE_PUNCH_OUTMASK[1]	= 2	Second interface output bit → Bit 2 SPIF2
MD26006 $MC_NIBBLE_PUNCH_INMASK[0]	= 1	First interface input bit → Bit 1 SPIF1
MD26006 $MC_NIBBLE_PUNCH_INMASK[1]	= 2	Second interface input bit → Bit 2 SPIF2

Automatically activated pre-initiation time

Dead times due to the reaction time of the punching unit can be minimized if the stroke can be initiated before the interpolation window of the axes is reached. The reference time for this is the interpolation end. The stroke is automatically initiated with `G603` and delayed by the set value in relation to the time that the end of interpolation is reached.

The delay time for stroke initiation can be adjusted in machine data:

MD26018 $MC_NIBBLE_PRE_START_TIME

Example:

With an IPO cycle of 5 ms, a stroke shall be released two cycles after reaching the interpolation end:

⇒ MD26018 $MC_NIBBLE_PRE_START_TIME = 0.01 [s]

A pre-initiation time can also be programmed in setting data:

SD42402 $SC_NIBPUNCH_PRE_START_TIME

This setting takes effective only if MD26018 = 0 has been set.

Monitoring of the input signal

If the "stroke active" signal is fluctuating between strokes due to plunger overshoots, for example, the message "undefined punching signal" can be also be output when interpolation is stopped.

The message output is dependent on the setting in machine data:

MD26020 $MC_NIBBLE_SIGNAL_CHECK

MD26020 = 0	No alarm
MD26020 = 1	Alarm

Minimum period between two strokes

A minimum time interval between two consecutive strokes can be programmed in setting data:

SD42404 $SC_MINTIME_BETWEEN_STROKES

Example:

There must be a minimum delay of at least 1.3 seconds between two stroke initiations irrespective of physical distance:

⇒ SD42404 $SC_MINTIME_BETWEEN_STROKES = 1.3 [s]

If a punching dwell time (`PDELAYON`) is also programmed, then the two times are applied additively.

If a pre-initiation time at `G603` is programmed, it will be effective only if the end of interpolation is reached before the time set in SD 42404:

The programmed time becomes operative immediately. Depending on the size of the block buffer, strokes that have already been programmed can be executed with this minimum interval. The following programming measures (example) can be taken to prevent this:

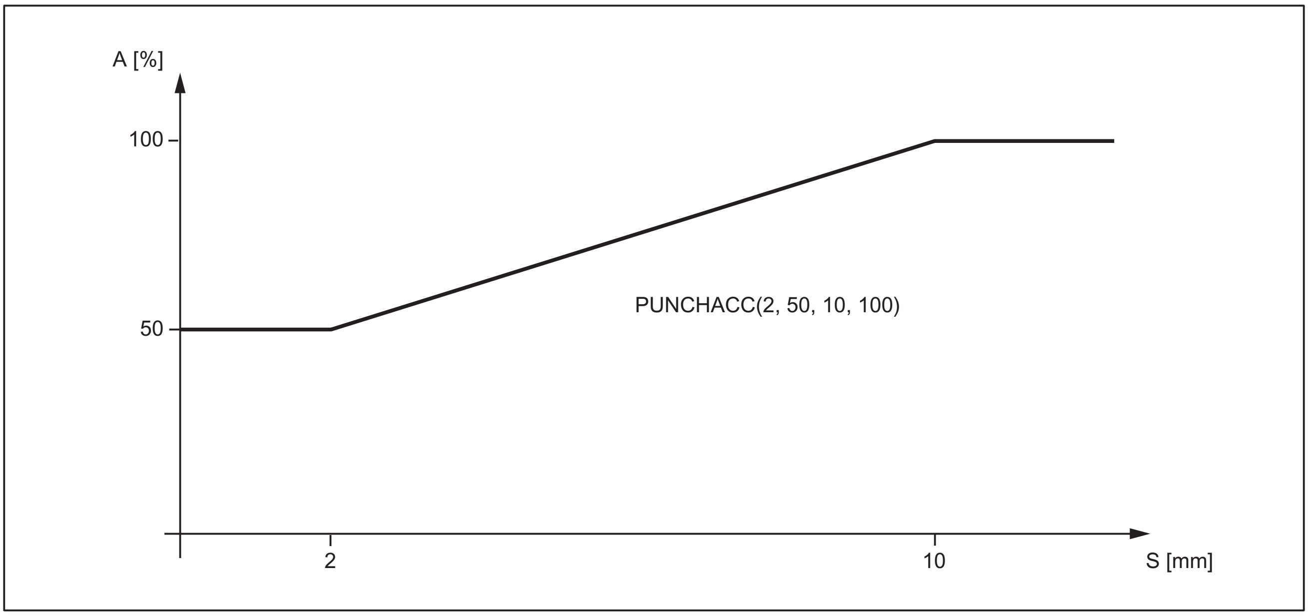
Program code
`N...`
`N100 STOPRE`
`N110 $SC_ MINTIME_BETWEEN_STROKES = 1.3`

The function is not active when SD42404 = 0.

Travel-dependent acceleration

An acceleration characteristic can be defined with `PUNCHACC (Smin, Amin, Smax, Amax)`. This command can be used to define different acceleration rates depending on the distance between holes.

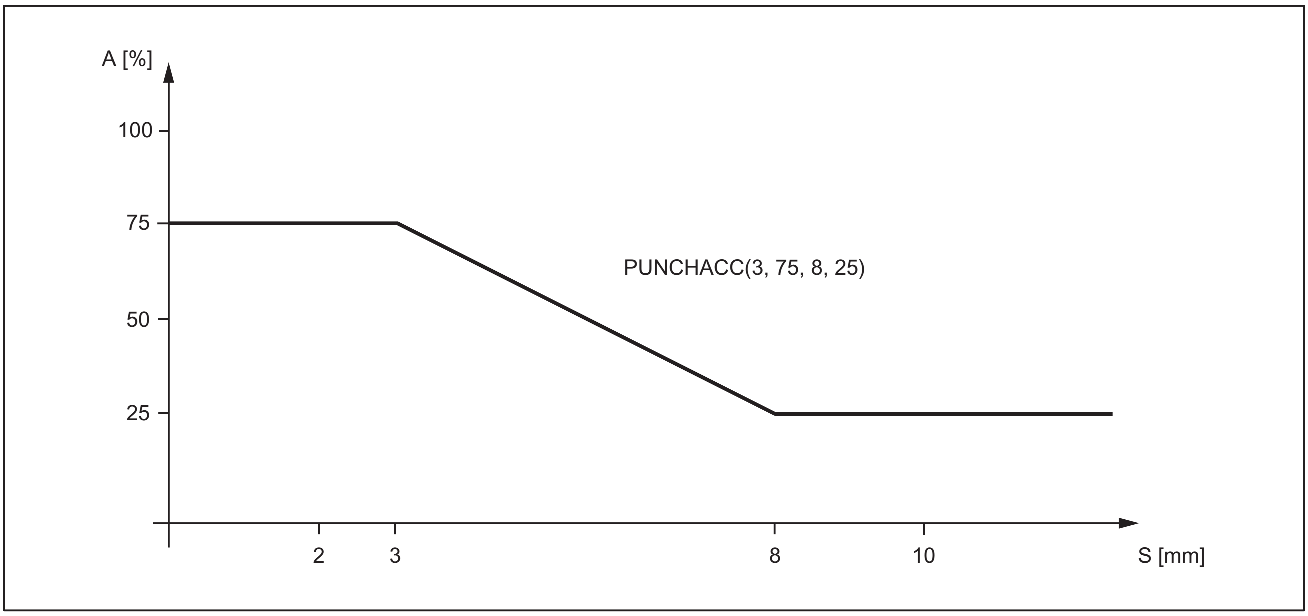
Example 1:



The characteristic defines the following acceleration rates:

Distance between holes	Acceleration
< 2 mm	The axis accelerates at a rate corresponding to 50 % of maximum acceleration.
2 - 10 mm	Acceleration is increased to 100 %, proportional to the spacing.
> 10 mm	The axis accelerates at the maximum rate (100%).

Example 2:



The characteristic defines the following acceleration rates:

Distance between holes	Acceleration
< 3 mm	The axis accelerates at a rate corresponding to 75 % of maximum acceleration.
3 - 8 mm	Acceleration is reduced to 25 %, proportional to the spacing.
> 10 mm	The axis accelerates at the maximum rate (25 %).

If a reduced acceleration rate has already been programmed via `ACC`, then the acceleration limits defined with `PUNCHACC` refer to the reduced acceleration rate.

The function is deselected with:

$S_{min} = S_{max} = 0$

The acceleration rate programmed beforehand with `ACC` remains operative.

Block search

In the case of a search for a block containing a nibbling function, it is possible to program whether the punch stroke is executed at the block beginning or suppressed.

The setting is programmed in machine data:

MD11450 $MN_SEARCH_RUN_MODE

Bit	Value	Meaning
5	0	Punch stroke at beginning of block is suppressed.
	1	Punch stroke at beginning of block is executed.

References:
Function Manual, Basic Functions; Mode Group, Channel, Program Operation, Reset Response (K1), Section:

9.3.3 Compatibility with earlier systems

Use of M functions

As in earlier versions, macro technology allows special M functions to be used instead of language commands (compatibility).

The M functions and equivalent language commands as used in earlier systems are as follows:

M20, M23	≙	SPOF
M22	≙	SON
M25	≙	PON
M26	≙	PDELAYON

Note

M functions can be configured in machine data.

When M functions are assigned to language commands, it must be noted that M functions are organized in auxiliary function groups.

Examples

`DEFINE M20 AS SPOF`	Punching/nibbling OFF
or `DEFINE M20 AS SPOF M=20`	Punching with auxiliary function output
`DEFINE M20 AS SPOF PDELAYOF`	Punching/nibbling OFF and punching with delay OFF
`DEFINE M22 AS SON`	Nibbling ON
or `DEFINE M22 AS SON M=22`	Nibbling ON with auxiliary function output
`DEFINE M25 AS PON`	Punching ON
or `DEFINE M25 AS PON M=25`	Punching ON with auxiliary function output
`DEFINE M26 AS PDELAYON`	Punching with delay ON
or `DEFINE M26 AS PDELAYON M=26`	Punching and auxiliary function output

Programming example:

```
Program code              Comment
:
:
N100 X100 M20             ; position without punch initiation
N110 X120 M22             ; Activate nibbling, before and after motion
                          ; stroke initiation
:
N120 X150 Y150 M25        ; Activate punching, initiate stroke at end
                          ; of the motion
:
:
```

9.4 Automatic path segmentation

9.4.1 General information

Function

One of the following two methods can be applied to automatically segment a programmed traversing path:

- Path segmentation with maximum path segment programmed via language command `SPP`
- Path segmentation with a number of segments programmed via language command `SPN`

Both functions generate sub-blocks independently.

In earlier systems

- `SPP<number>` corresponds to `E<number>`
- `SPN<number>` corresponds to `H<number>`

Since addresses `E` and `H` now represent auxiliary functions, language commands `SPP` and `SPN` are used to avoid conflicts. The new procedure is therefore not compatible with those implemented in earlier systems. Both language commands (`SPP` and `SPN`) can be configured.

Note

The values programmed with `SPP` are either mm or inch settings depending on the initial setting (analogous to axes).

The automatic path segmentation function ensures that the path is divided into equidistant sections with linear and circle interpolation.

When the program is interrupted and automatic path segmentation is active (`SPP`/`SPN`), the contour can be reentered only at the beginning of the segmented block. The first punch stroke is executed at the end of this sub-block.

`SPP` and `SPN` can be activated only if geometry axes are configured.

SPP

The automatic path segmentation function `SPP` divides the programmed traversing path into sections of equal size according to the segment specification.

The following conditions apply:

- Path segmentation is active only when `SON` or `PON` is active.

  (Exception: MD26014 $MC_PUNCH_PATH_SPLITTING = 1)
- `SPP` is modally active, i.e. the programmed segment remains valid until it is programmed again, but it can be suppressed on a block-by-block (non-modal) basis by means of `SPN`.

- The path segments are rounded off by the control system if required so that a total programmed distance can be divided into an integral number of path sections.
- The path segment unit is either mm/stroke or inch/stroke (depending on axis settings).
- If the programmed `SPP` value is greater than the traversing distance, then the axis is positioned on the programmed end position without path segmentation.
- `SPP = 0`, reset or program end delete the programmed `SPP` value. The `SPP` value is not deleted when punching/nibbling is deactivated.

SPN

The automatic path segmentation function `SPN` divides the traversing path into the programmed number of path segments.

The following conditions apply:

- Path segmentation is active only when `SON` or `PON` is active.

  (Exception: MD26014 $MC_PUNCH_PATH_SPLITTING = 1)
- `SPN` has a non-modal action.
- Any previously programmed `SPP` value is suppressed for the block containing SPN, but is re-activated again in the following blocks.

Supplementary conditions

- The path segmentation function is operative with linear and circular interpolation.

  The interpolation mode remains unchanged, i.e. circles are traversed when circular interpolation is selected.
- If a block contains both `SPN` (number of strokes) and `SPP` (stroke path), then the number of blocks is activated in the current block while the stroke path is activated in all blocks that follow.
- Path segmentation is active only in conjunction with punching or nibbling functions.

  (Exception: MD26014 $MC_PUNCH_PATH_SPLITTING = 1).
- Any programmed auxiliary functions are output before, during the first or after the last sub-block.
- In the case of blocks without traversing information, the rules which govern the programming of `SON` and `PON` also apply to `SPP` and `SPN`. In other words, a stroke is initiated only if an axis motion has been programmed.

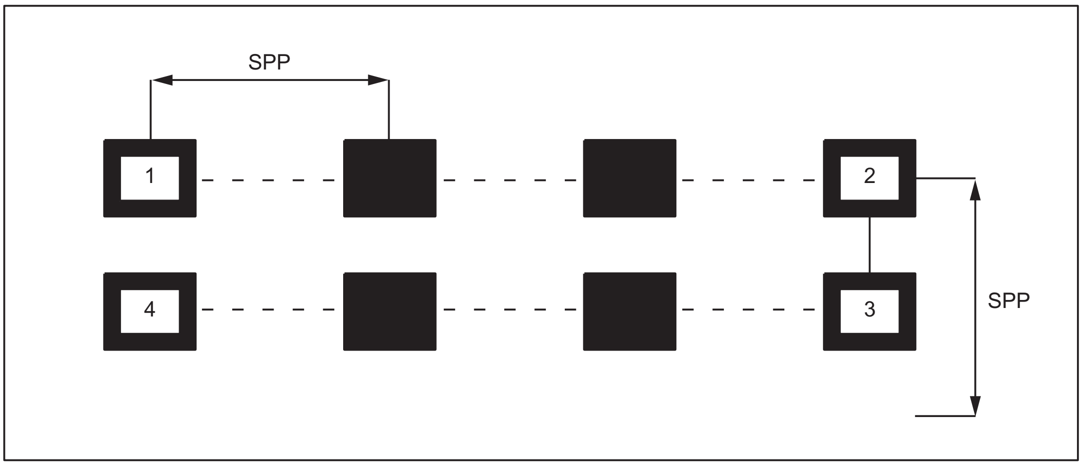
9.4.2 Operating characteristics with path axes

MD26010

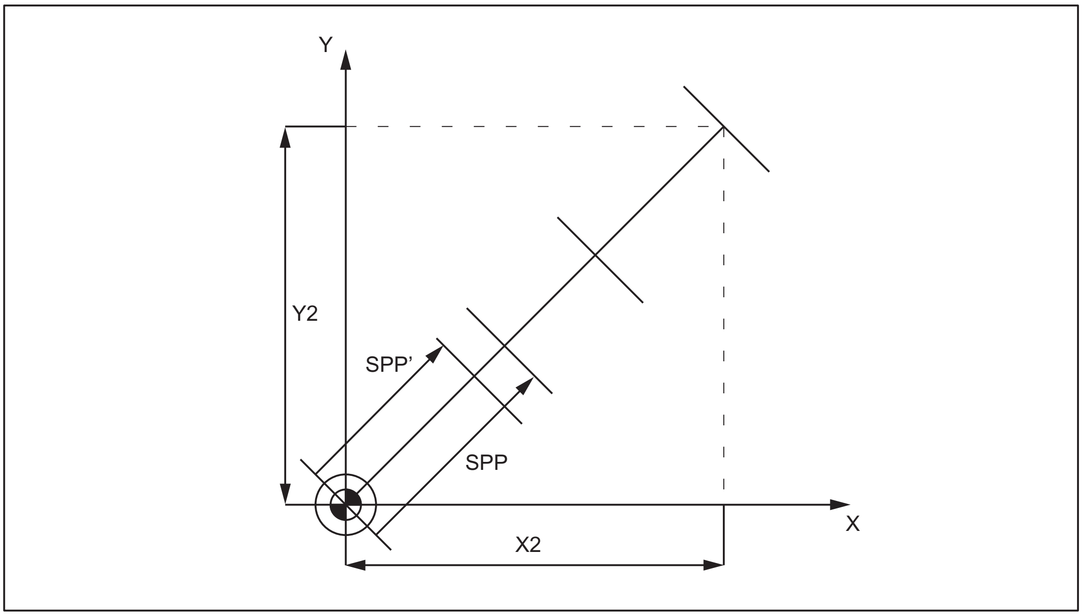
All axes defined and programmed via machine data:
MD26010 $MC_PUNCHNIB_AXIS_MASK
are traversed along path sections of identical size with SPP and SPN until the programmed end point is reached. This also applies to rotatable tool axes if programmed. The response can be adjusted for single axes.

Example of SPP

Program code	Comment
N1 G01 X0 Y0 SPOF	; Position without punch initiation
N2 X75 SPP=25 SON	; Nibble with feed value 25 mm; initiate punch
	; before the first movement and after each
	; path section,
:	
:	
N3 Y10	; Position with reduced SPP value, because
:	; traversing distance < SPP value, and initiate punch
	; after the movement.
:	
:	
N4 X0	; Reposition with punch initiation
	; after each path section.
:	



If the programmed path segmentation is not an integral multiple of the total path, then the feed path is reduced.



X2/Y2:	Programmed traversing distance
SPP:	Programmed SPP value
SPP':	Automatically rounded-off offset distance

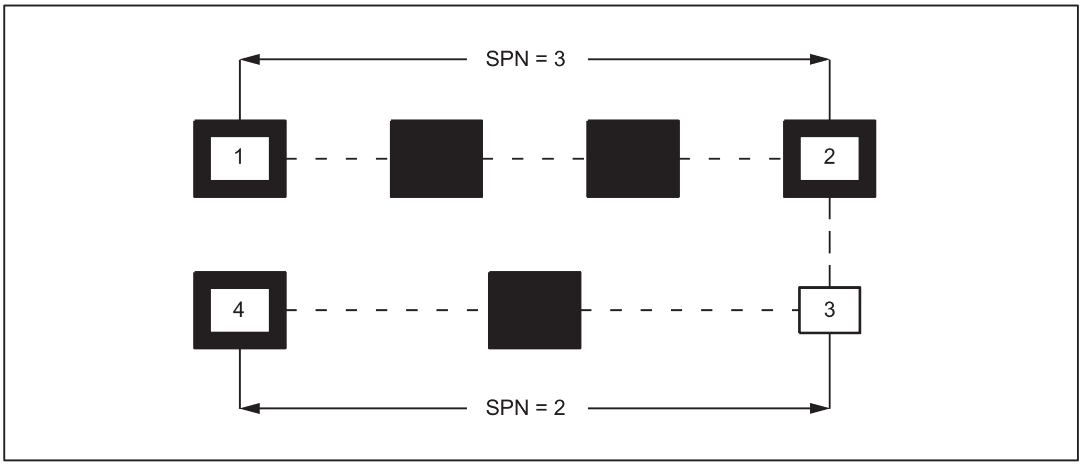
Figure 9-4 Path segmentation

Example of SPN

The number of path segments per block is programmed via `SPN`.

A value programmed via `SPN` takes effect on a non-modal basis for both punching and nibbling applications. The only difference between the two modes is with respect to the first stroke. This is normally executed at the beginning of the first segment with nibbling operations and at its end with punching operations. This means that when n segments are programmed, n strokes are executed with punching operations but n+1 with nibbling. Furthermore, where no travel information is available, only a single stroke is executed, even if several are programmed. Should it be necessary to generate several strokes at one position, then the corresponding number of blocks without traversing information must be programmed.

Program code	Comment
N1 G01 X0 Y0 SPOF	; position without punch initiation
N2 X75 SPN=3 SON	; Activate nibbling. The total path is
	; divided into three segments. Before the first
	; movement and at the end of each segment, a
	; stroke is triggered.
:	
:	
:	
:	
N3 Y10 SPOF	; Position without punch initiation
N4 X0 SPN=2 PON	; activate punching. The total path is divided into 2
	; path segments. Because punching has been activated,
	; the first stroke is initiated at the end of the first path segment.
	;
:	
:	
:	
:	



Example

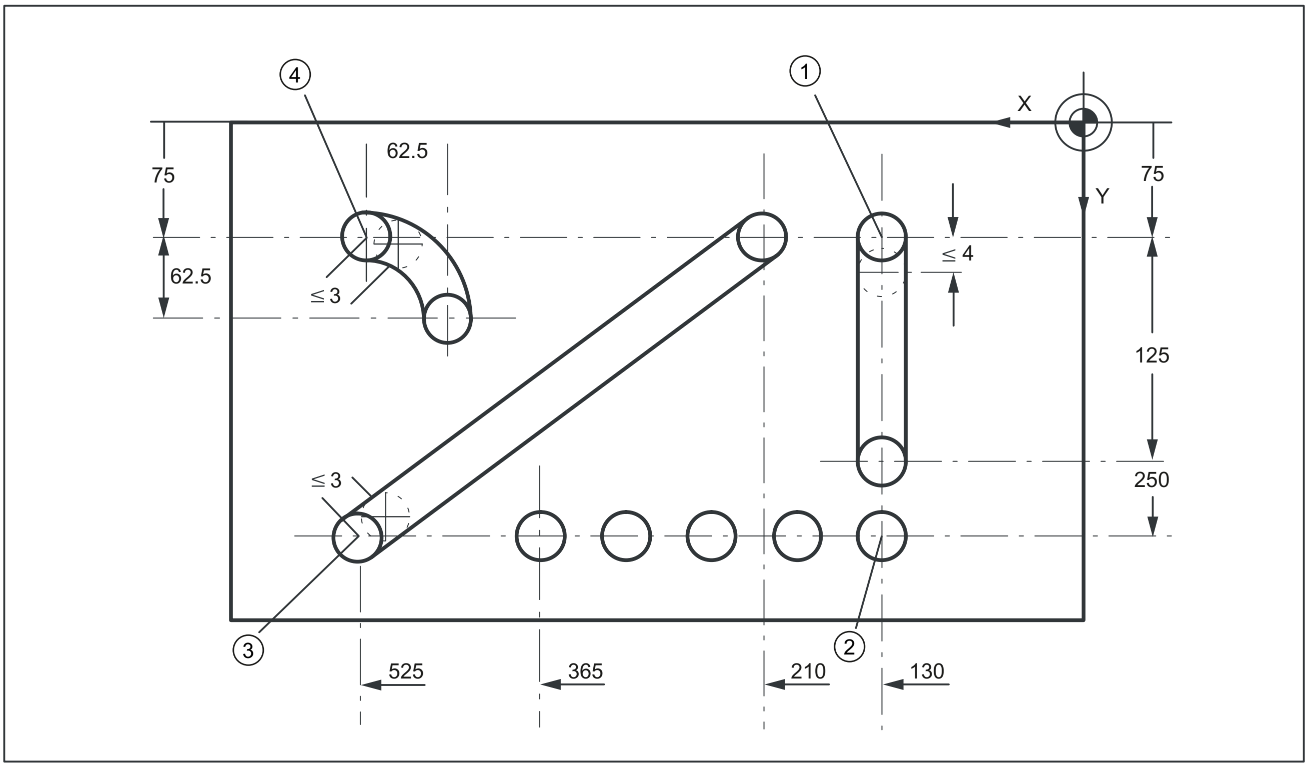


Figure 9-5 Workpiece

Extract from program

Program code	Comment
N100 G90 X130 Y75 F60 SPOF	; Positioning to starting point (1) of the ; vertical nibbling path sections
N110 G91 Y125 SPP=4 SON	; End point coordinates (incremental) ; Path segment: 4 mm, activate nibbling
N120 G90 Y250 SPOF	; Absolute dimensioning, position at ; the start point (2) of the horizontal ; nibbling path section
N130 X365 SPN=4 SON	; End point coordinates, four path sections, ; activate nibbling
N140 X525 SPOF	; Positioning to starting point (3) of the ; oblique nibbling path
N150 X210 Y75 SPP=3 SON	; End point coordinates path segment: 3 mm, ; activate nibbling
N160 X525 SPOF	; Positioning to starting point (4) of the ; nibbling path on pitch circle path

Program code	Comment
N170 G02 G91 X-62.5 Y62.5 I0 J62.5 SON	; Incremental circular interpolation with
	; interpolation parameters, nibbling
	; activating
N180 G00 G90 Y300 SPOF	; Positioning

9.4.3 Response in connection with single axes

MD26016

The path of single axes programmed in addition to path axes is distributed evenly among the generated intermediate blocks as standard.

In the following example, the additional rotary axis C is defined as a synchronous axis.

If this axis is programmed additionally as a "Punch-nibble axis":
MD26010 $MC_PUNCHNIB_AXIS_MASK = 1,
, then the behavior of the synchronous axis can be varied as a function of machine data:
MD26016 $MC_PUNCH_PARTITION_TYPE
.

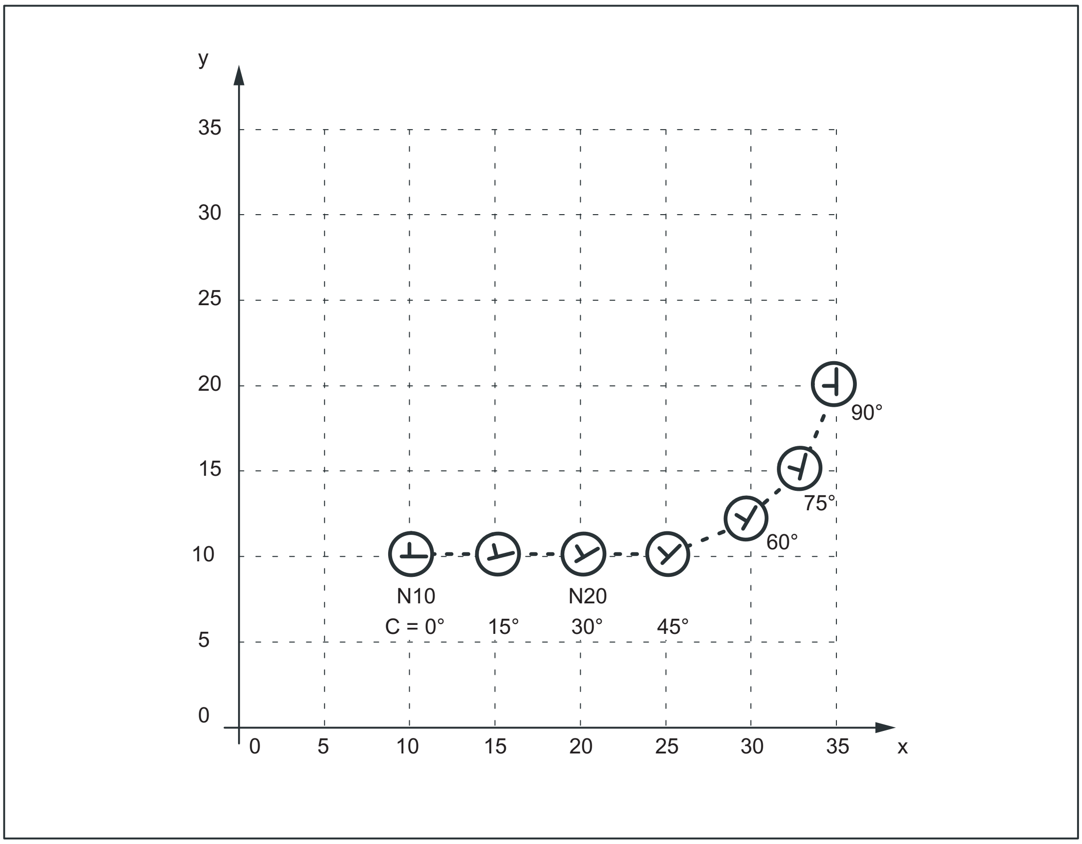
Programming example:

Program code
N10 G90 G1 PON X10 Y10 C0 F10000
N20 SPP=5 X25 C45
N30 G3 SPN=3 X35 Y20 I0 J10 C90

MD26016 $MC_PUNCH_PARTITION_TYPE=0 (default setting)

With this setting, the axes behave as standard, i.e. the programmed special axis motions are distributed among the generated intermediate blocks of the active path segmentation function in all interpolation modes.

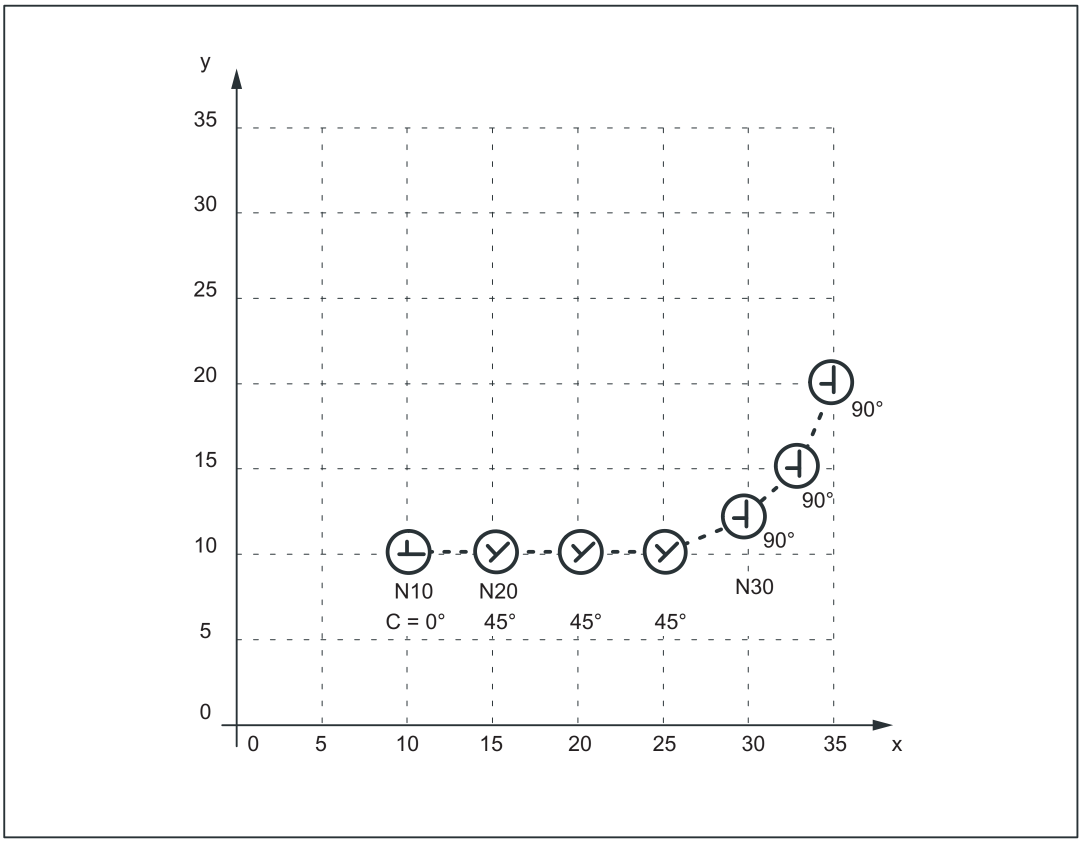
In block `N20`, the C axis is rotated through 15° in each of the three intermediate blocks. The axis response is the same in block `N30`, in the case of circular interpolation (three sub-blocks, each with 15° axis rotation).



MD26016 $MC_PUNCH_PARTITION_TYPE=1

In contrast to the behavior described above, here the synchronous axis travels the entire programmed rotation path in the first sub-block of the selected path segmentation function.

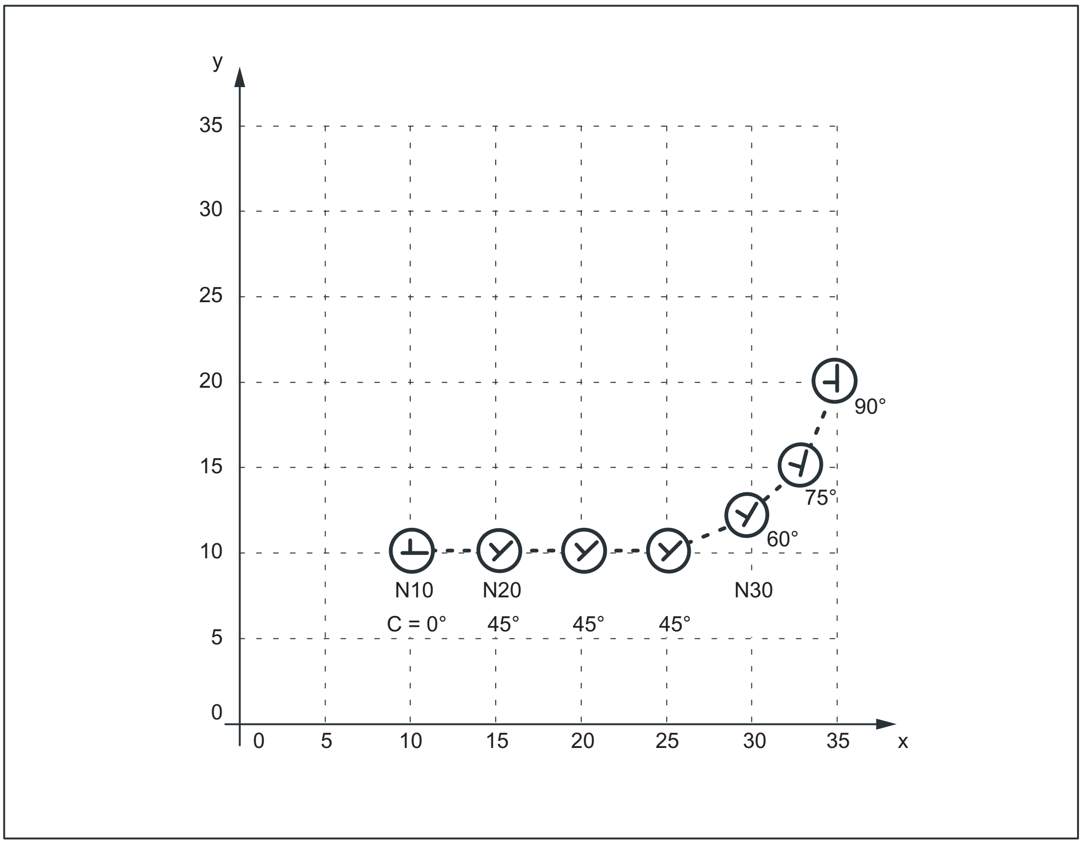
Applied to the example, the C axis already reaches the programmed end position `C=45` when it reaches X position X=15. It behaves in the same way in the circular interpolation block below.



MD26016 $MC_PUNCH_PARTITION_TYPE=2

MD26016=2 is set in cases where the axis must behave as described above in linear interpolation mode, but according to the default setting in circular interpolation mode (see 1st case).

The axis behavior for the example is then as follows: In block `N20`, the C axis is rotated to C=45° in the first sub-block. The following circular interpolation block rotates the C axis through 15° in every sub-block.



The axis response illustrated in the diagram above can be particularly useful when applied to the axis of a rotatable tool in cases where it is used to place the tool in a defined direction (e.g. tangential) in relation to the contour, but where the tangential control function **must not** be applied. However, it is not a substitute for the tangential control function since the start and end positions of the rotary axis must always be programmed.

Note

Additional offset motions of special axes (in this case, rotary axis C) are implemented via a zero offset.

Supplementary conditions

- If the C axis is not defined as a "Punch-nibble axis", then the C axis motion path is not segmented in block `N30` in the above example nor is a stroke initiated at the block end.
- If the functionality described above is to be implemented in a variant not specific to nibbling applications, but with alignment of the special axis, then stroke initiation can be suppressed by the following PLC interface signal:

  DB 21, 22 DBX3.2 (stroke suppression)

  (Application: e.g. alignment of electron beam during welding)

  A similar response can be programmed with the following machine data setting:

  MD26014 $MC_PUNCH_PATH_SPLITTING=1

  In this case, the path is segmented irrespective of punching or nibbling functions.

9.5 Rotatable tool

9.5.1 General information

Function overview

The following two functions are provided for nibbling/punching machines with rotatable punch and lower die:

- Coupled motion

  for synchronous rotation of punch and die
- Tangential control

  for normal alignment of rotary axes for punches in relation to workpiece

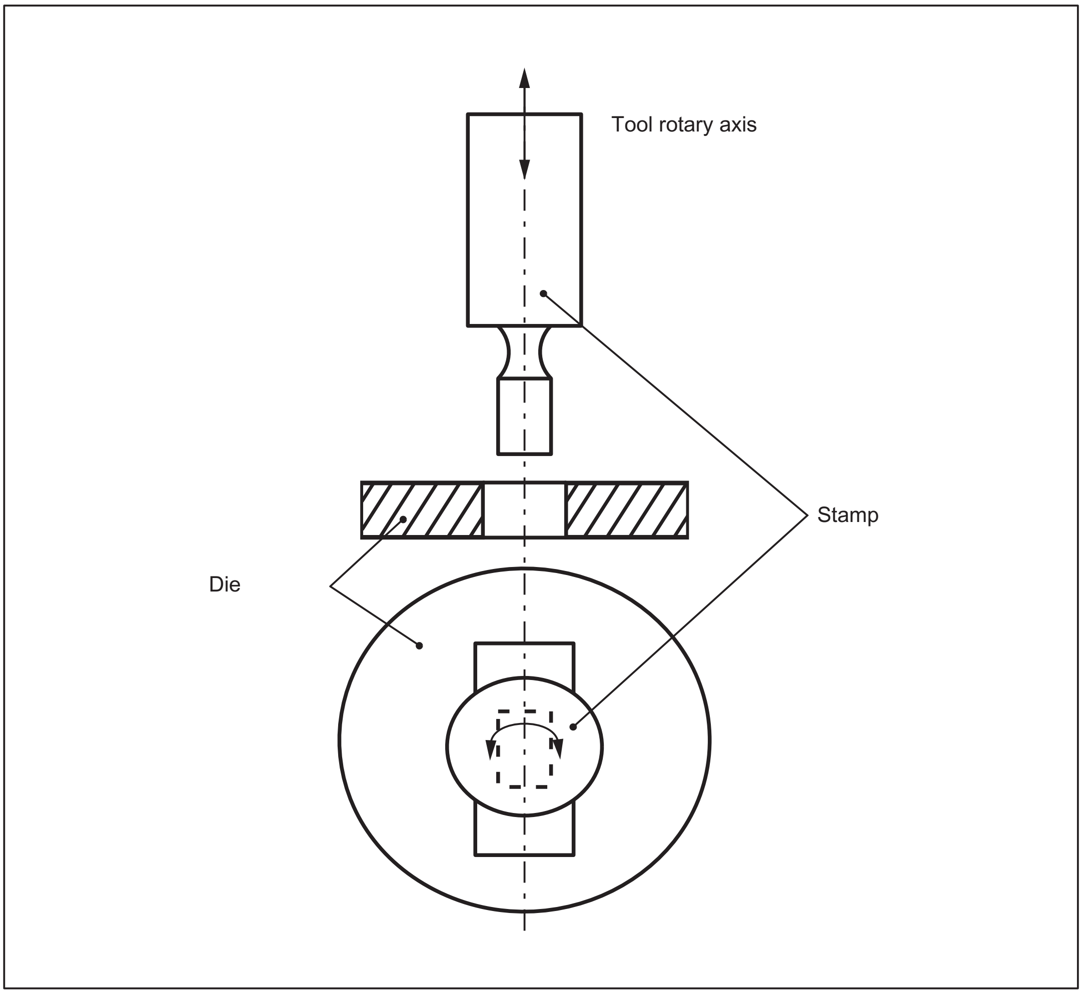


Figure 9-6 Illustration of a rotatable tool axis

9.5.2 Coupled motion of punch and die

Function

Using the standard function "Coupled motion", it is possible to assign the axis of the die as a coupled motion axis to the rotary axis of the punch.

Activation

The "Coupled motion" function is activated or deactivated with language commands `TRAILON` and `TRAILOF` respectively.

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

Example

Example of a typical nibbling machine with rotatable punches where C is the punch axis and C1 the die axis:

Program code	Comment
:	
:	
TRAILON (C1, C, 1)	; Enabling the coupled-motion grouping
G01 X100 Y100 C0 PON	; Initiate stroke with C axis /
	; C1 axis position C=0=C1
X150 C45	; Initiate stroke with C axis /
	; C1 axis position C=45=C1
:	
:	
M30	

Basic position

No coupled-motion groupings are active after power up. Once the two tool axes have approached the reference point, the coupled-motion grouping is not generally separated again.

This can be achieved by:

- Program single activation of the coupled-motion grouping (see example above)
- Program MD setting:

  MD20110 $MC_RESET_MODE_MASK, bit 8=1

In this way, the coupled-motion grouping remains active after RESET/part program start or end.

9.5.3 Tangential control

Function

The rotary tool axes on punching/nibbling machines are aligned tangentially to the programmed path of the master axes by means of the "Tangential control" function.

Activation

The "Tangential control" function is activated and deactivated with language commands `TANGON` and `TANGOF` respectively.

References:
Programming Manual, Production Planning

Mode of operation

The tangential axis is coupled to the interpolation of the master axes. It is therefore not possible to position the axis at the appropriate punching position tangentially to the path independently of velocity. This may lead to a reduction in machining velocity if the dynamics of the rotary axis are unfavorable in relation to those of the master axes. Additional offset angles can be programmed directly via language command `TANGON`.

Note

If the tool (punch and die) is positioned by two separate drives, then the functions "Tangential control" and "Coupled motion" can be used.

Notice: The "Tangential control" function must be activated first followed by "Coupled motion".

The tangential control function automatically aligns the punch vertically to the direction vector of the programmed path. The tangential tool is positioned before the first punching operation is executed along the programmed path. The tangential angle is always referred to the positive X axis. A programmed additional angle is added to the calculated angle.

The tangential control function can be used in the linear and circular interpolation modes.

Example: Linear interpolation

The punching/nibbling machine has a rotatable punch and die with separate drives.

Programming example:

Program code	Comment
:	
:	
N2 TANG (C, X, Y, 1, "B")	; Definition of leading and following axes,
	; C is the slave axis for X and Y in the
	; basic coordinate system
N5 G0 X10 Y5	; Start position
N8 TRAILON (C1, C, 1)	; Activate coupled motion of rotatable
	; tool axes C/C1
N10 Y10 C225 PON F60	; C/C1 axis rotates to 225° → stroke
N15 X20 Y20 C45	; C/C1 axis rotates to 45° → stroke
N20 X50 Y20 C90 SPOF	; C/C1 axis rotates to 90°, no
	; stroke initiation
N25 X80 Y20 SPP=10 SON	; Path segmentation: four strokes are performed
	; with tool rotated to 90°
N30 X60 Y40 SPOF	; Positioning
N32 TANGON (C, 180)	; Activate tangential control,
	; offset angle of rotatable tool axes 180°
N35 X30 Y70 SPN=3 PON	; Path segmentation, three strokes for active
	; tangential control and an
	; offset angle of 180°
N40 G91 C45 X-10 Y-10	; C/C1 rotates to 225° (180° + 45° INC),
	; Tangential control deactivated because no
	; path segmentation → stroke
N42 TANGON (C, 0)	; Tangential control without offset
N45 G90 Y30 SPN=3 SON	; Path segmentation, three strokes for active
	; Tangential control without offset angle
N50 SPOF TANGOF	; Deactivate stroke initiation +
	; Tangential control
N55 TRAILOF (C1, C)	; Activate coupled motion of rotatable
	; tool axes C/C1
N60 M2	

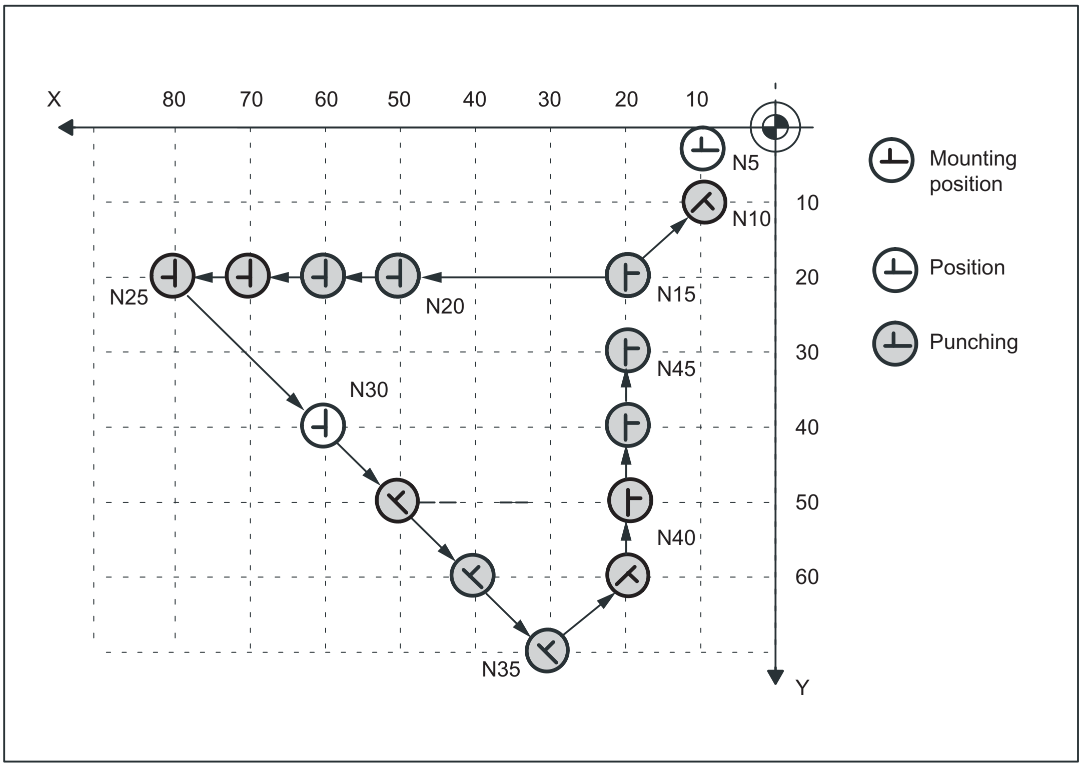


Figure 9-7 Illustration of programming example in XY plane

Example: Circular interpolation

In circular interpolation mode, particularly when path segmentation is active, the tool axes rotate along a path tangentially aligned to the programmed path axes in each sub-block.

Programming example:

Program code	Comment
:	
:	
N2 TANG (C, X, Y, 1, "B")	; Definition of leading and following axes,
	; C is the slave axis for X and Y in the
	; basic coordinate system
N5 G0 F60 X10 Y10	; Start position
N8 TRAILON (C1, C, 1)	; Activate coupled motion of the
	; rotatable tool axes C/C1
	; for lower and upper tool
N9 TANGON (C, -90)	; Activate tangential control
	; with offset 270°
N10 G02 X30 Y30 I20 J0 SPN=2 PON	; Circular interpolation with path segmentation,
	; 2 strokes are executed with 270°

Program code	Comment
	; Offset angle and tangential
	; alignment along the circular path
N15 G0 X70 Y10 SPOF	; Positioning
N17 TANGON (C, 90)	; Activate tangential control
	; with offset 90°
N20 G03 X35,86 Y24,14 CR=20 SPP=16 SON	; Circular interpolation, path segmentation, 4
	; strokes are performed with 90°
	; offset angle and tangential
	; alignment along the circular path
N25 G0 X74.14 Y35.86 C0 PON	; Rotation of the tool axes to
	; 0°, stroke
N27 TANGON (C, 0)	; Activate tangential control
	; with offset 0°
N30 G03 X40 Y50 I-14,14 J14,14 SPN=5 SON	; Circular interpolation, path segmentation,
	; 5 strokes with 0° offset angle
	; and tangential alignment on
	; the circular path
N35 G0 X30 Y65 C90 SPOF	; Position without active
	; tangential control
N40 G91 X-10 Y-25 C180	; Positioning, C axis rotates to 270°
N43 TANGOF	; Deactivate tangential control
N45 G90 G02 Y60 I0 J10 SPP=2 PON	; Circular interpolation, path segmentation,
	; Two strokes without tangential control
	; with C=270°
N50 SPOF	; Punching OFF
N55 TRAILOF (C1, C)	; Deactivate coupled motion of the
	; rotatable tool axes C/C1
N60 M2	

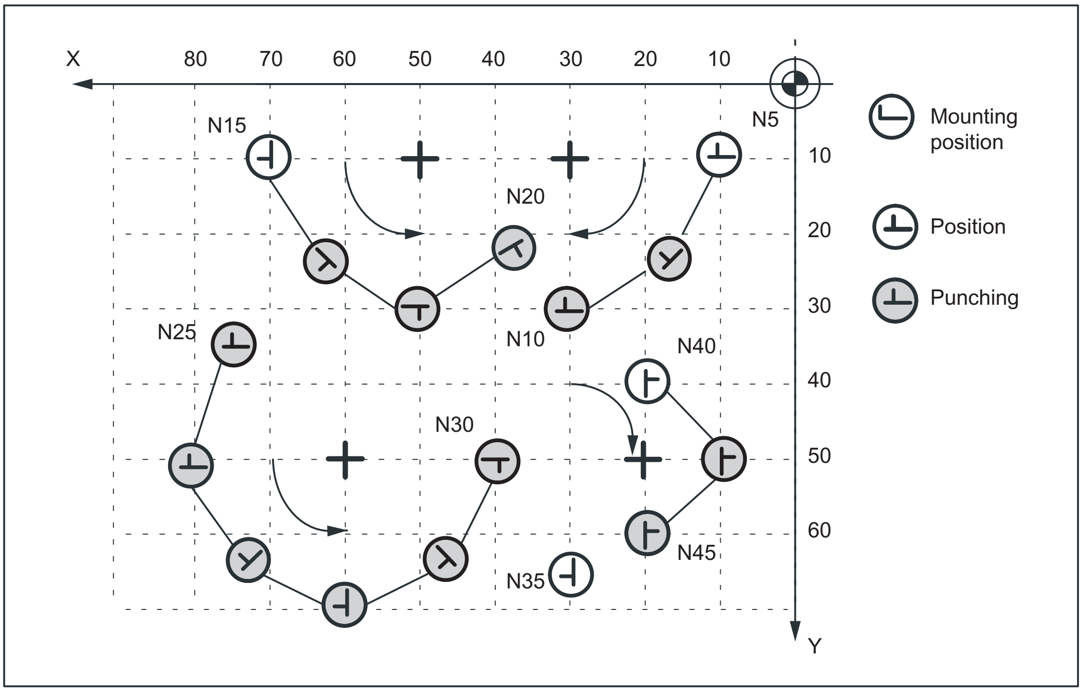


Figure 9-8 Illustration of programming example in XY plane

9.6 Protection zones

Clamping protection zone

The "clamping protection zone" function is contained as a subset in the "Protection zones" function. Its purpose is to simply monitor whether clamps and tool could represent a mutual risk.

Note

No by-pass strategies are implemented for cases where the clamp protection is violated.

References:
Function Manual, Basic Functions; Axis Monitoring, Protection Zones (A3)

9.7 Supplementary conditions

Availability of function "Punching and nibbling"

The function is an option ("Punching and nibbling functions"), which must be assigned to the hardware through the license management.

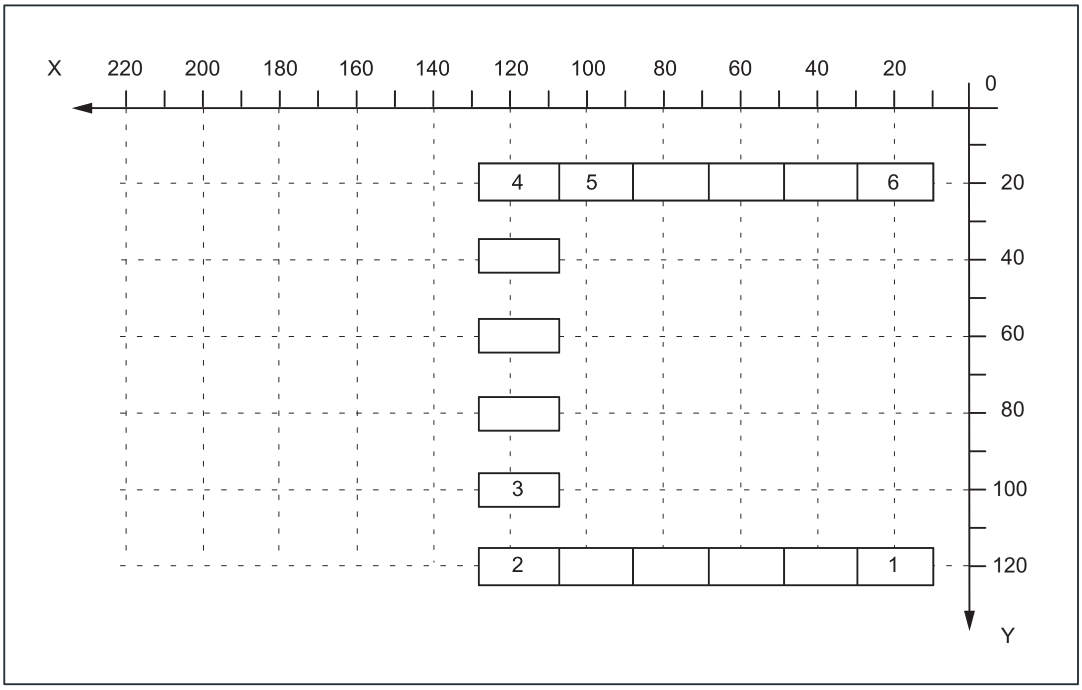
9.8 Examples

9.8.1 Examples of defined start of nibbling operation

Example 1

Example of defined start of nibbling operation

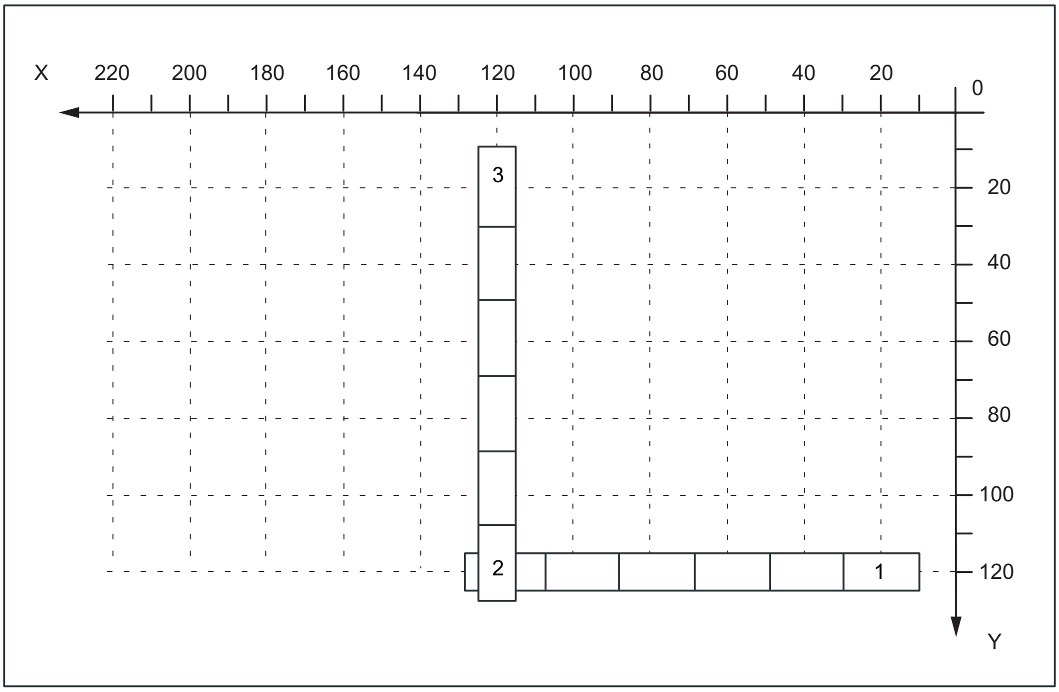
Program code	Comment
:	
:	
N10 G0 X20 Y120 SPP= 20	; Position 1 is approached
N20 X120 SON	; Defined start of nibbling, first stroke at
	; "1", last stroke at "2"
N30 Y20	; Defined start of nibbling, first stroke at
	; "3", last stroke at "4"
N40 X20	; Defined start of nibbling, first stroke at
	; "5", last stroke at "6"
N50 SPOF	
N60 M2	



Example 2

This example utilizes the "Tangential control" function. Z has been selected as the name of the tangential axis.

Program code	Comment
:	
:	
N5 TANG (Z, X, Y, 1, "B")	; Define tangential axis
N8 TANGON (Z, 0)	; Select tangential control
N10 G0 X20 Y120	; Position 1 is approached
N20 X120 SPP=20 SON	; Defined start of nibbling,
	; tangential control selected,
	; first stroke at "1", last stroke at "2"
N30 SPOF TANGOF	; Deselect nibbling mode and deselect
	; tangential control
N38 TANGON (Z, 90)	; Select tangential control
N40 Y20 SON	; Defined start of nibbling,
	; tangential control selected,
	; first stroke at "2" rotated 90 degrees to
	; block N20, last stroke at "3"
N50 SPOF TANGOF	; Deselect nibbling mode and deselect
	; tangential control
N60 M2	



Examples 3 and 4 for defined start of nibbling

Example 3: Programming of SPP

Program code	Comment
:	
:	
N5 G0 X10 Y10	; Positioning
N10 X90 SPP=20 SON	; Defined start of nibbling,
	; 5 punches initiated
N20 X10 Y30 SPP=0	; One punch is initiated at the end of the path
N30 X90 SPP=20	; 4 punches initiated at intervals of 20 mm
N40 SPOF	
N50 M2	

Example 4 Programming of SPN

Program code	Comment
:	
:	
N5 G0 X10 Y10	; Positioning
N10 X90 SPN=4 SON	; Defined start of nibbling, 5
	; 5 punches initiated
N20 X10 Y30 PON	; One punch is initiated at the end of the path
N30 X90 SPN=4	; 4 punches initiated
N40 SPOF	
N50 M2	

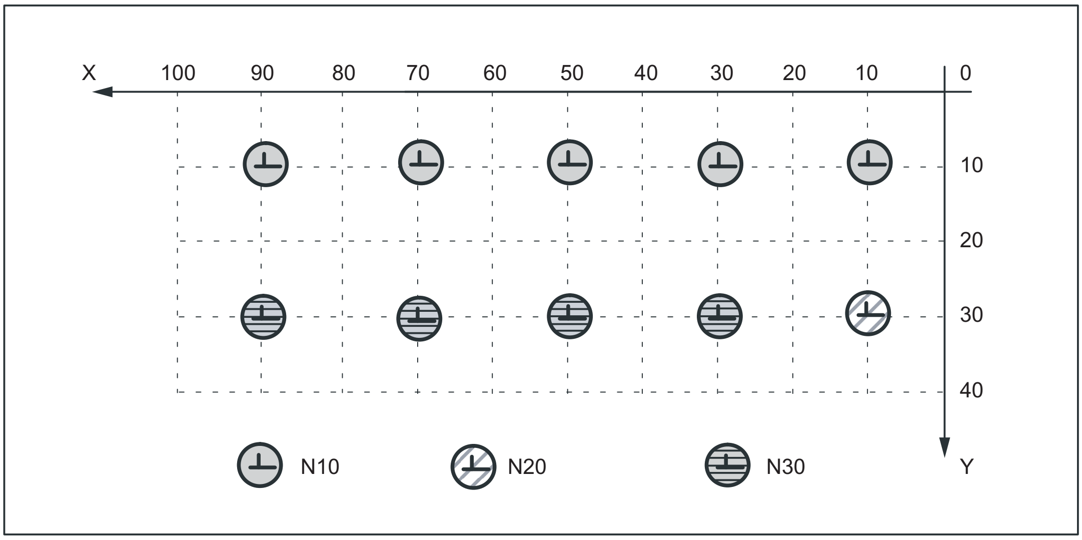


Figure 9-9 Examples 3 and 4 for defined start of nibbling

Examples 5 and 6 without defined start of nibbling

Example 5 Programming of SPP

Program code	Comment
:	
:	
N5 G0 X10 Y30	; Positioning
N10 X90 SPP=20 PON	; No defined start of nibbling,
	; 4 punches initiated
N15 Y10	; One punch is initiated at the end of the path
N20 X10 SPP=20	; 4 punches initiated at intervals of 20 mm
N25 SPOF	
N30 M2	

Example 6 Programming of SPN

Program code	Comment
:	
:	
N5 G0 X10 Y30	; Positioning
N10 X90 SPN=4 PON	; No defined start of nibbling,
	; 4 punches initiated
N15 Y10	; One punch is initiated at the end of the path
N20 X10 SPN=4	; 4 punches initiated
N25 SPOF	
N30 M2	

Example 7 Application example of SPP programming

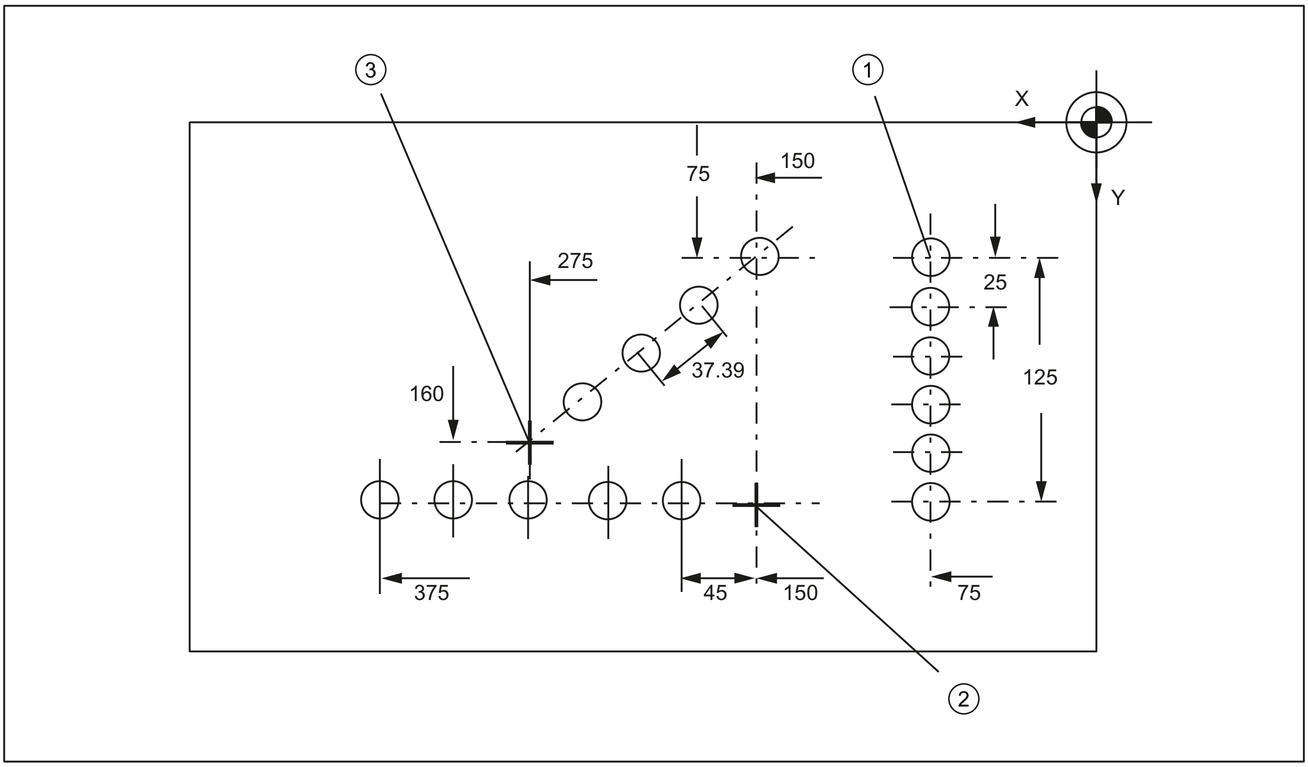


Figure 9-10 Workpiece

Extract from program:

Program code	Comment
N100 G90 X75 Y75 F60 PON	; Positioning to starting point (1) of the
	; vertical line of holes, punch single hole
N110 G91 Y125 SPP=25 PON	; End point coordinates (incremental),
	; path segment: 25 mm, activate punching
N120 G90 X150 SPOF	; Absolute dimensioning, position at
	; the start point (2) of the horizontal line of holes
N130 X375 SPP=45 PON	; End point coordinates, path segment: 45 mm
N140 X275 Y160 SPOF	; Positioning to starting point (3) of the
	; oblique line of holes
N150 X150 Y75 SPP=40 PON	; End point coordinates, programmed
	; path segment: 40 mm, calculated
	; path segment: 37.39 mm
N160 G00 Y300 SPOF	; Positioning

9.9 Data lists

9.9.1 Machine data

9.9.1.1 General machine data

Number	Identifier: $MN_	Description
11450	SEARCH_RUN_MODE	Block search parameter settings

9.9.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES[n]	Reset G groups
26000	PUNCHNIB_ASSIGN_FASTIN	Hardware assignment for input-byte with stroke control
26002	PUNCHNIB_ASSIGN_FASTOUT	Hardware assignment for output-byte with stroke control
26004	NIBBLE_PUNCH_OUTMASK[n])	Mask for quick output bits
26006	NIBBLE_PUNCH_INMASK[n]	Mask for quick input bits
26008	NIBBLE_PUNCH_CODE[n]	Determination of the M functions
26010	PUNCHNIB_AXIS_MASK	Definition of punching and nibbling axes
26012	PUNCHNIB_ACTIVATION	Activation of punching and nibbling functions
26014	PUNCH_PATH_SPLITTING	Activation of automatic path segmentation
26016	PUNCH_PARTITION_TYPE	Behavior of single axes with active automatic path segmentation
26018	NIBBLE_PRE_START_TIME	Automatically activated pre-initiation time
26020	NIBBLE_SIGNAL_CHECK	Monitoring of the input signal

9.9.2 Setting data

9.9.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42400	PUNCH_DWELL_TIME	Dwell time
42402	NIBPUNCH_PRE_START_TIME	Pre-start time
42404	MINTIME_BETWEEN_STROKES	Minimum time interval between two consecutive strokes

9.9.3 Signals

9.9.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
No stroke enable	DB21, ...DBX3.0	-
Manual stroke initiation	DB21, ...DBX3.1	-
Stroke suppression	DB21, ...DBX3.2	-
Stroke inoperative	DB21, ...DBX3.3	-
Delayed stroke	DB21, ...DBX3.4	-
Manual stroke initiation	DB21, ...DBX3.5	-

9.9.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Stroke initiation active	DB21, ...DBX38.0	-
Acknowledgement of manual stroke initiation	DB21, ...DBX38.1	-

9.9.4 Language commands

G group	Language command	Meaning	
35	`SPOF`	Stroke / Punch OFF	Punching and nibbling OFF
35	`SON`	Stroke ON	Nibbling ON
35	`SONS`	Stroke ON	Nibbling ON (position controller)
35	`PON`	Punch ON	Punching ON
35	`PONS`	Punch ON	Punching ON (position controller)
36	`PDELAYON`	Punch with Delay ON	Punching with delay ON
36	`PDELAYOF`	Punch with Delay OFF	Punching with delay OFF
Path segmentation			
	`SPP`		Path per stroke, modal action
	`SPN`		Number of strokes per block, non-modal action

P2: Positioning axes 10

10.1 Product brief

Axes for auxiliary movements

In addition to axes for machining a workpiece, modern machine tools can also be equipped with axes for auxiliary movements, e.g.:

- Axis for tool magazine
- Axis for tool turret
- Axis for workpiece transport
- Axis for pallet transport
- Axis for loader (also multi-axis)
- Axis for tool changer
- Axis for sleeve assembly / end support

The axes for the workpiece machining are called path axes. Within the channel they are guided by the interpolator such that they start simultaneously, accelerate, reach the end point and stop together.

Axes for auxiliary movements are traversed independently of the path axes at a separate, axis-specific feedrate. In the past, many of these axes were moved hydraulically and started by an auxiliary function in the part program. With the closed-loop axis control implemented in the NC, the axis can be addressed by name in the part program and its actual position displayed on the screen.

Note

"Positioning axis/Auxiliary spindle" option

Axes for auxiliary movements must not be interpolating ("full-value") NC axes. Auxiliary movements may also be carried out using special axes, which can be obtained using the "Positioning axis/Auxiliary spindle" option.

Functional restrictions

Optional positioning axes/auxiliary spindles have fewer functions. The following functions are **not** possible:

- Using the axis as a path axis, geometry axis, or special path axis
- Incorporating the axis into the geometry axis grouping (GEOAX)
- Rigid thread cutting and tapping

Commissioning

As standard, axes are defined as interpolating axes:

MD30460 $MA_BASE_FUNCTION_MASK bit 8 = 0

If an axis is to be operated as a positioning axis/auxiliary spindle with reduced functionality, the value for bit 8 must be set to "1":

MD30460 $MA_BASE_FUNCTION_MASK bit 8 = **1**

Function

The "positioning axes" function makes it easier to integrate axes for auxiliary movement into the control system:

- during programming:

  The axes are programmed together with the axes for workpiece machining in the same part program, without having to sacrifice valuable machining time.

  There are special (`POS`, `POSA`) traversing instructions.

- during program testing/start-up:

  Program testing and start-up are performed simultaneously for all axes.

- during operation:

  Operation and monitoring of the machining process commence simultaneously for all axes.

- during PLC configuring/commissioning:

  No allowance has to be made on PLC or external computers (PCs) for synchronization between axes for machining and axes for auxiliary movements.

- during system configuring:

  A second channel is not required.

Motions and interpolations

Each channel has one path interpolator and at least one axis interpolator with the following interpolation functions:

- for path interpolator:

  Linear interpolation (G1), circular interpolation (G2 / G3), spline interpolation, etc.
- for axis interpolator:

  If a positioning axis is programmed, an axis interpolator starts in the control (with linear interpolation G1).
- End-of-motion criterion:

  The programmed end position of a positioning axis has been reached when the end-of-motion criterion FINEA, COARSA or IPOENDA is fulfilled.
- Path axes with rapid traverse movement:

  Path axes can be traversed in linear or non-linear interpolation mode with rapid traverse movement (G0).
- Autonomous singleaxis operations:

  Single PLC axes, command axes started via static synchronized actions or asynchronous reciprocating axes can be interpolated independently of the NCK.

  An axis/spindle interpolated by the main run then reacts independently of the NC program. The channel response triggered by the program run is decoupled to transfer the control of a certain axis / spindle to the PLC.
- Control by PLC:

  All channel-specific signals normally act to the same extent on path and positioning axes.

  Positioning axes can be controlled via additional, axis-specific signals.

  PLC axes are traversed by the PLC via special function blocks in the basic program; their movements can be asynchronous to all other axes. The travel motions are executed separate from the path and synchronized actions.

10.2 Own channel, positioning axis or concurrent positioning axis

When axes are provided for auxiliary movements on a machine tool, the required properties will decide whether the axis is to be:

- programmed in an internal part program (see Section "Own channel - only 840D sl (Page 602)").
- programmed in the same part program as the machining operation (see Section "Positioning axis (Page 602)").
- started exclusively by the PLC during machining (see Section "Concurrent positioning axis (Page 605)").

10.2.1 Own channel - only 840D sl

A channel represents a self-contained NC which, with the aid of a part program, can be used to control the movement of axes, spindles and machine functions independently of other channels.

Non-dependence between channels

Independence between channels is assured by means of the following provisions:

- An active part program per channel
- Channelspecific interface signals such as
  - DB21, ... DBX7.1 (NC start)
  - DB21, ... DBX7.3 (NC stop)
  - DB21, ... DBX7.7 (reset)
- One feedrate override per channel
- One rapid traverse override per channel
- Channelspecific evaluation and display of alarms
- Channelspecific display, e.g. for
  - Actual axis positions
  - Active G functions
  - Active auxiliary functions
  - Current program block
- Channel-specific testing and channel-specific control of programs:
  - Single block
  - Dry run (DRY RUN)
  - Block search
  - Program test

References

For more information on the channel functionality, please refer to:
function manual, Basic Functions; BAG, Channel, Program Operation, Reset Response (K1)

10.2.2 Positioning axis

Positioning axes are programmed together with path axes, i.e. with the axes that are responsible for workpiece machining.

Commands for positioning axes and path axes can be included in the same NC block. Although they are programmed in the same NC block, the path and positioning axes are not interpolated together and do not reach their end point simultaneously (no direct time relationship, see also Section "Motion behavior and interpolation functions (Page 606)").

Positioning axis types and block change

The block change time depends on the programmed positioning axis type (refer also to Section "Block change (Page 621)"):

Type	Description
1	The block change occurs when all path and positioning axes have reached their programmed end point.
2	The block change occurs when all path axes have reached their programmed end point. With positioning axis type 2, it is possible to approach the programmed end point across several block limits.
3	It is possible to set the block change within the braking ramp of the single axis interpolation if the criteria for the motion end and the block change are fulfilled for the path interpolation.

Motion synchronization

Positioning axes permit movements to be activated from the same machining program and such movements to be synchronized at block limits (type 1) or at explicit points by means of a `WAITP` command (type 2).

Motion end criterion for block change in the brake ramp

For single-axis interpolation, it is also possible to set another end-of-motion criterion for the block change in the braking ramp.

Traverse path axes in G0 as positioning axis

Each path axis can be traversed as positioning axis in rapid traverse movement (`G0`). Thus all axes travel to their end point independently.

In this way, two sequentially programmed X and Z axes are treated like positioning axes in conjunction with `G0`. The block change to axis Z can be initiated by axis X as a function of the braking ramp time setting (100-0%). Axis Z starts to move while axis X is still in motion. Both axes approach their end point independently of one another.

Axis types

Positioning axes can be linear axes and rotary axes.

Positioning axes can also be configured as indexing axes.

Independence of positioning axes and path axes

The mutual independence of positioning and path axes is ensured by the following measures:

- No shared interpolation
- Each positioning axis has a dedicated axis interpolator
- Dedicated feed override for each positioning axis
- Dedicated programmable feedrate
- Dedicated "axis-specific delete distance-to-go" interface signal

Dependencies

Positioning axes are dependent in the following respects:

- A shared part program
- Starting of positioning axes only at block boundaries in the part program
- With rapid traverse movement `G0` path axes traverse as positioning axes in one of two different modes
- No rapid traverse override
- The following interface signals act on the entire channel and therefore on positioning axes:
  - DB21, ... DBX7.1 (NC start)
  - DB21, ... DBX7.3 (NC stop)
  - DB21, ... DBX7.7 (reset)
  - DB21, ... DBX6.1 (read-in disable)
- Alarms specific to program and channel also deactivate positioning axes.
- Program control (dry run feed, program test, DRF, ... etc.) also act on positioning axes
- Block search and single block also act on positioning axes.
- The last block with a programmed end-of-motion criterion that was processed in the search run serves as a container for setting all axes.
- Group 1 (modal movement commands) of the G functions `G0`, `G1`, `G2`, ...) does not apply to positioning axes.

References:
Programming Manual Fundamentals.

Applications

The following are typical applications for positioning axes:

- Single-axis loaders
- Multi-axis loaders without interpolation (PTP → point-to-point traversing)
- Workpiece feed and transport

Other applications are also possible:

- With `G0` workpiece delivery and workpiece transport can travel to their end points independently of one another.
- On machines with several machining processes in sequence: Significant reduction in individual machining steps due to block change in the braking ramp of the single-axis interpolation.

> **Note**
>
> Positioning axes are not suitable for multi-axis loaders that require interpolation between the axes (path interpolator).

10.2.3 Concurrent positioning axis

Concurrent positioning axes are positioning axes with the following properties:

- Activation from the PLC need not take place at block limits, but can be implemented at any time in any operating mode (even when a part program is already being processed in the channel).
- Program command `WAITP` is required to move a concurrent positioning axis from the part program immediately after power ON.
- The part program continues to run uninhibited, even if the concurrent positioning axis has not reached the position defined by the PLC.
- An automatic axis change is possible, depending on the setting in the machine data MD30552 $MA_AUTO_GET_TYPE.
- With programming commands:
  - `GET(<axis>)` or `WAITP(<axis>)` becomes a concurrent positioning axis of the channel axis again.
  - `"RELEASE (axis)"` or `WAITP(<axis>)` is a channel axis that becomes a concurrent axis under PLC control.

Activation from PLC

For SINUMERIK 840D sl, the concurrent positioning axis is activated via FC 18 from the PLC.

- Feedrate

  For feedrate = 0, the feedrate is determined from the following machine data:

  MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)
- Absolute dimensions (`G90`), incremental dimensions (`G91`)

  Absolute dimensions along shortest path for rotary axes (`<rotary axis name>=DC(<value>)`)

The following functions are defined:

- Linear interpolation (`G1`)
- Feedrate in mm/min or degrees/min (`G94`)
- Exact stop (`G9`)
- Settable zero offsets currently selected are valid

Applications

Typical applications for concurrent positioning axes include:

- Tool magazines with manual loading and unloading during machining
- Tool magazines with tool preparation during machining

10.3 Motion behavior and interpolation functions

10.3.1 Path interpolator and axis interpolator

Path interpolator

Every channel has a path interpolator for a wide range of interpolation modes such as linear interpolation (`G1`), circular interpolation (`G2`/`G3`), spline interpolation etc.

Axis interpolator

Each channel has axis interpolators in addition to path interpolators. The maximum number corresponds to the maximum number of existing channel axes.

If a positioning axis is programmed, an axis interpolator starts in the control with straight line interpolation `G1`. This axis interpolator runs independently of the path interpolator until the programmed end position of the positioning axis has been reached.

There is no time relationship between the path interpolator and the axis interpolator, nor between the axis interpolators.

Path control mode (`G64`) is not possible with positioning axes.

The programmed end position of a positioning axis has been reached when the end-of-motion criterion `FINEA`, `COARSA` or `IPOENDA` is fulfilled.

10.3.2 Interpolation response of path axis in G0

Path axes can be traversed in linear or non-linear interpolation mode in rapid traverse movement (G0).

Linear interpolation

Properties:

- The path axes are interpolated together.
- The tool movement programmed with `G0` is executed at the highest possible speed (rapid traverse).
- The rapid traverse velocity is defined separately for each axis in the following machine data:

  MD32000 $MA_MAX_AX_VELO
- If the rapid traverse movement is executed simultaneously on several axes, the rapid traverse speed is determined by the axis which requires the most time for its section of the path.

Linear interpolation is always performed in the following cases:

- For a G-code combination with `G0` that does **not** allow positioning axis motion, e.g.:

  `G40`, `G41`, `G42`, `G96`, `G961` and MD20750 $MC_ALLOW_G0_IN_G96 == FALSE
- With a combination of `G0` with `G64`
- When a compressor or transformation is active
- In point-to-point (PTP) travel mode
- When a contour handwheel is selected (FD=0)
- In case of an active frame with rotation of geometry axes
- If nibbling is active for geometry axes

Non-linear interpolation

Properties:

- Each path axis interpolates as a single axis (positioning axis) independently of the other axes at the rapid traverse velocity defined in the following machine data:

  MD32000 $MA_MAX_AX_VELO
- The channel-specific delete distance-to-go command via the PLC and synchronized actions is applied to all positioning axes that were programmed as path axes.

In non-linear interpolation, with reference to the axial jerk:

- The setting of the concerned positioning axes BRISKA, SOFTA, DRIVEA

or

- The setting in the machine data:

  MD32420 $MA_JOG_AND_POS_JERK_ENABLE

  and

  MD32430 $MA_JOG_AND_POS_MAX_JERK

The existing system variables which refer to the distance-to-go ($AC_PATH, $AC_PLTBB and $AC_PLTEB) are supported.

⚠ CAUTION
Risk of collision
As traversal of another contour is possible with non-linear interpolation, synchronized actions that refer to coordinates of the original path may not be active.

Selection of interpolation type

The interpolation type that should be effective for G0 is adjusted with the following machine data:

MD20730 $MC_G0_LINEAR_MODE (interpolation response in G0)

Value	Meaning
0	In the rapid traversing mode (`G0`) the **non-linear** interpolation is active. Path axes are traversed as positioning axes.
1	In the rapid traversing mode (`G0`) the **linear** interpolation is active. The path axes are interpolated together.

The desired interpolation response in `G0` can also be programmed via the two following part program commands, independently of the default:

`RTLIOF`	Deactivating the linear interpolation. ⇒ In the rapid traversing mode (G0), the **non-linear** interpolation is active.
`RTLION`	Activating the linear interpolation. ⇒ In the rapid traversing mode (G0), the **linear** interpolation is active.

The currently set interpolation response of the path axes with `G0` can be queried with system variable $AA_G0MODE.

Note

In both interpolation types, rapid override is channel-specific.

10.3.3 Autonomous singleaxis operations

Functionality

Single PLC axes, command axes started via static synchronized actions or asynchronous reciprocating axes can be interpolated independently of the NCK. An axis/spindle interpolated by the main run then reacts independently of the NC program with respect to:

- NC stop
- Alarm handling
- Program control
- End of program
- RESET

Boundary conditions

Axes/spindles currently operating according to the NC program are not controlled by the PLC.

Command axis movements **cannot** be started via non-modal or modal synchronized actions for PLC-controlled axes/spindles. Alarm 20143 is signaled.

Transfer axis control to the PLC

Description of the sequence

1. PLC → NCK: Request to control the axis
   DB31, ... DBX28.7 = 1 (PLC controls axis)
2. NCK: Checks whether the axis is a main run axis or a neutral axis.
3. NCK: Checks whether an additional axis may be controlled from the PLC.
4. NCK confirms the transfer:
   - DB31, ... DBX63.1 = 1 (PLC controls the axis)
   - System variable $AA_SNGLAX_STAT = 1

Result: The PLC controls the axis/spindle.

Alternatives

Initial state: The axis is controlled by the PLC. As a result of a channel stop, the channel is in the "interrupted" state.

- Axis state "inactive" ⇒
  - The stop state is canceled.
  - If the axis is started, this directly results in axis motion.
- Axis state "active" ⇒
  - The stop state is **not** canceled.
  - Generate the axis state according to **Use case 2 "Stop axis"**.
  - Resume axis motion according to **Use case 3 "Continue axis motion"**.
- A reset is performed in the channel ⇒

  This process is asynchronous to control acceptance by the PLC. The two previously mentioned alternatives can occur or the axis is assigned to the channel and is reset.

Boundary conditions

Axes/spindles, traversed by an NC program, cannot be transferred to the PLC. Axes/spindles, which are traversed by static synchronized actions or as oscillating axis, as neutral axis, concurrent positioning axis or command axis, can be transferred.

Relinquish axis control by the PLC

Description of the sequence:

1. PLC → NCK: The PLC returns axis control to the NCK

   DB31, ... DBX28.7 = 0 (PLC controls axis)
2. NCK: Checks whether an axial alarm is present.
3. NCK: Checks whether a movement has been activated that has still not been completed? If yes, then the movement is stopped with an axial stop according to **Use case 2 "Stop axis/spindle"**.
4. NCK: Carries out an axial reset corresponding to **Use case 4 "Reset axis/spindle"**.
5. NCK confirms the acceptance:

   DB31, ... DBX63.0 = 0 (reset executed)

   DB31, ... DBX63.1 = 0 (PLC controls the axis)

   DB31, ... DBX63.2 = 0 (axis stop active)

   System variable $AA_SNGLAX_STAT = 0

Result: The NCK has now taken over control of the axis/spindle.

Alternatives

In the following cases the NCK confirms the transfer - but internally sets the "stopped" channel state for the axis/spindle:

- The channel is in the "interrupted" state
- A stop alarm is active for the channel
- A stop alarm is active for the mode group

Boundary conditions

The axis/spindle must be operating under PLC control.

The NCK confirms acceptance of an axis/spindle only if an axial alarm is not active.

Description of the sequence based on use cases

Requirement

The axis/spindle is controlled by the PLC

Relevant NC/PLC interface signals

One of the axes/spindles controlled by the PLC can be influenced by the following NC/PLC interface signals independent of the NC program:

- DB21, ... DBX6.2 (delete distance-to-go)
- DB31, ... DBX28.1 (reset)
- DB31, ... DBX28.2 (continue)
- DB31, ... DBX28.6 (stop along braking ramp)

For signal flow between the NCK and PLC at the NC/PLC interface during autonomous single operations, see Section "Control by the PLC (Page 629)".

Use case 1: Cancel axis/spindle

The behavior when canceling the axis/spindle function is the same as for "delete distance-to-go":

DB21, ... DBX6.2 = 1 (delete distance-to-go)

Use case 2: Stop axis/spindle

The following traversing motion of the axis/spindle controlled from the main run is stopped:

- PLC axis
- Asynchronous oscillating axis
- Command axis by static synchronized action
- Overlaid motion: $AA_OFF,DRF handwheel traversal, online tool offset and external zero offset.

Following axis movements of the axis/spindle are not stopped.

Description of the sequence:

- PLC → NCK: Request to stop the axis/spindle
  DB31, ... DBX28.6 = 1 (stop along braking ramp)
- NCK: Brakes the axis along a ramp.
- NCK confirms the execution:
  - DB31, ... DBX60.6 = 1 (exact stop coarse)
  - DB31, ... DBX60.7 = 1 (exact stop fine)
  - DB31, ... DBX63.2 = 0 (axis stop active)
  - DB31, ... DBX64.6 / 7 = 0 (traversing command minus/plus)
  - Axis status interrupted with system variable $AA_SNGLAX_STAT == 3

Result: The axis/spindle is stopped.

Note

Following axis movements

Following axis movements can only be suppressed when the leading axis stops.

Retraction motion

Retraction motion triggered by the "Extended stop and retract" function cannot be stopped.

References

Function Description, Special Functions; Extended Stop and Retract (R3)

Use case 3: Continue axis/spindle

Traversing motion interrupted after **Use case 2 "Stop axis"** should be continued.

Description of the sequence:

- PLC → NCK: Continue axis
  DB31, ... DBX28.2 = 1 (continue)
- NCK: Checks whether for the axis/spindle an axial alarm with delete criterion "CANCELCLEAR" or "NCSTARTCLEAR" is present? If yes, then this is deleted.
- NCK: Checks whether axis motion can be resumed? If yes, then the axis/spindle makes the transition into the "active" state.
- NCK confirms the execution:
  - DB31, ... DBX60.6 = 0 (exact stop coarse)
  - DB31, ... DBX60.7 = 0 (exact stop fine)
  - DB31, ... DBX63.2 = 0 (axis stop active)
  - DB31, ... DBX64.6 / 7 = 1 (traversing command minus/plus)
  - Axis state active with system variable $AA_SNGLAX_STAT == 4.

Result: Traversing motion of the axis/spindle is continued.

Boundary conditions

In the following cases, the request to continue is ignored:

- The axis/spindle is not controlled by the PLC.
- The axis/spindle is not in the stopped state.
- An alarm is pending for the axis/spindle.

Use case 4: Reset axis/spindle (reset)

Description of the sequence:

- PLC → NCK: Reset request for this axis/spindle
  DB31, ... DBX28.1 = 1 (reset)
- NCK: Transitions the axis/spindle into the "stopped" state.
- NCK: Interrupts the stopped sequences and signals to the PLC the interruption - essentially the same as for "Delete distance-to-go".
- NCK: The internal states for the axis/spindle are reset.
- NCK: The axial machine data effective at reset becomes active.

Note

In contrast to a reset due to DB31, ... DBX28.1 = 1 (reset), in conjunction with a channel reset, no axial machine data is active for axes controlled from the PLC.

- NCK confirms the execution:
  - DB31, ... DBX63.0 = 1 (reset executed)
  - DB31, ... DBX63.2 = 0 (axis stop active)
  - System variable $AA_SNGLAX_STAT = 1
- NCK: Ends this operation.

10.3.4 Autonomous single-axis functions with NC-controlled ESR

Extended stop numerically controlled

The numerically controlled extended stop and retract function is also available for single axes and is configurable with axial machine data:

Delay time for ESR single axis with

MD37510 $MA_AX_ESR_DELAY_TIME1

ESR time for interpolatory braking of the single axis with

MD37511 $MA_AX_ESR_DELAY_TIME2

The values of these axial machine data are however effective only if the axis/spindle is a single axis.

The NC-controlled extended stop and retract is activated by the axial trigger $AA_ESR_TRIGGER[axis]. It works analogously to $AC_ESR_TRIGGER and applies exclusively to single axes.

References:

Function Manual, Special Functions; Coupled axes and ESR (M3)

Extended retract numerically controlled

For retracting single axes, the value must have been programmed via POLFA(axis, type, value) and the following conditions must be met:

- The axis must be a single axis at the time of triggering
- $AA_ESR_ENABLE[axis]=1
- POLFA(axis, type, value) with type=1 or type=2 only

  POLFA(axis, value, axis, type, axis type).

Note

NC-controlled extended stop for single axes:

The trigger is only effective if the axis is a single axis at the time of triggering, otherwise the trigger is ignored and the axial stop for this axis is **not** executed.

NC-controlled extended retract for single axes:

The channel-specific NC-controlled extended retract function is **not** effective for single axes. All axes that are single axes at the time of triggering $AC_ESR_TRIGGER will be ignored for channel-specific retraction.

This also applies when all the parameters for retraction are set, such as:

MD37500 $MA_ESR_REACTION

$AA_ESR_ENABLE for the axis, etc.

Examples

Extended **stopping** of a single axis:

MD37500 $MA_ESR_REACTION[AX1]=22

MD37510 $MA_AX_ESR_DELAY_TIME1[AX1]=0.3

MD37511 $MA_AX_ESR_DELAY_TIME2[AX1]=0.06

...

$AA_ESR_ENABLE[AX1] = 1

$AA_ESR_TRIGGER[AX1]=1 ; axis begins to stop the process here

Extended **retraction** of a single axis:

MD37500 $MA_ESR_REACTION[AX1]=21

...

$AA_ESR_ENABLE[AX1] = 1

POLFA(AX1, 1, 20.0); AX1 is assigned the axial retraction position 20.0 ; (absolute)

$AA_ESR_TRIGGER[AX1]=1 ; AX1 begins to retract here

POLFA(axis, type): permissible programming abbreviation

POLFA(axis, 0/1/2) ; quick deactivation/activation

> ⚠ WARNING
>
> **No preprocessing limitation**
>
> If abbreviated notation is used and only the type is changed, make sure that the value for the retraction position or retraction path in the application is meaningful!
>
> The abbreviated notation should only be used in exceptional circumstances.
>
> This particularly applies after:
>
> A power on, the retraction path or the retraction position must be reset.
>
> POLFA(axis, 1, $AA_POLFA[axis]) ; causes a preprocessing stop
>
> POLFA(axis, 1); does **not** cause a preprocessing stop

10.4 Positioning axis dynamic response

Velocity

The positioning axes traverse at the axis-specific feedrate programmed for them. As described under "Motion behavior and interpolation functions (Page 606)", this feedrate is not influenced by the path axes.

The feedrate is programmed as an axis-specific velocity in units of min/mm, inch/min or degrees/min.

The axis-specific feedrate is always permanently assigned to a positioning axis by the axis name.

If a positioning axis has no programmed feedrate, the control system automatically applies the rate set in axis-specific machine data:

MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)

The programmed axis-specific feedrate is active until the end of the program.

Feedrate override

The path and positioning axes have separate feedrate overrides. Each positioning axis can be adjusted by its own axis-specific feedrate override.

Rapid traverse override

Rapid traverse override applies only to path axes. Positioning axes have no rapid traverse interpolation (only axial linear interpolation G01) and therefore no rapid traverse override.

Revolutional feedrate

In JOG mode the behavior of the axis/spindle also depends on the setting of SD41100 JOG_REV_IS_ACTIVE (revolutional feedrate when JOG active).

- If this setting data is active, an axis/spindle is always moved with revolutional feedrate MD32050 $MA_JOG_REV_VELO (revolutional feedrate with JOG) or MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feedrate with JOG with rapid traverse overlay) as a function of the master spindle.
- If the setting data is not active, the behavior of the axis/spindle depends on SD43300 $SA_ASSIG_FEED_PER_REV_SOURCE (revolutional feedrate for positioning axes/spindles)
- If the setting data is not active, the behavior of a geometry axis on which a frame with rotation is effective depends on the channel-specific setting data SD42600 $SC_JOG_FEED_PER_REV_SOURCE. (In the operating mode JOG, revolutional feedrate for geometry axes on which a frame with rotation is effective.)

Maximum axial acceleration

With positioning axis motions, one of the two following maximum values is effective depending on the set positioning axis dynamic response mode:

- MD32300 $MA_MAX_AX_ACCEL [0] (maximum axial acceleration for path motions in the dynamic response mode DYNNORM)
- MD32300 $MA_MAX_AX_ACCEL [1] (maximum axial acceleration for path motions in the dynamic response mode DYNPOS)

The positioning axis dynamic response mode is set in the NC-specific machine data:

MD18960 $MN_POS_DYN_MODE = <mode>

<mode>	Meaning
0	Effective maximum axial acceleration: MD32300 $MA_MAX_AX_ACCEL[0]
1	Effective maximum axial acceleration: MD32300 $MA_MAX_AX_ACCEL[1]

Maximum axial jerk

When traversing positioning axes with active jerk limitation, the value from one of the following machine data takes effect as maximum axial jerk:

- MD32430 $MA_JOG_AND_POS_MAX_JERK (maximum axial jerk for positioning axis motions)
- MD32431 $MA_MAX_AX_JERK [0] (maximum axial jerk for path motions in the dynamic response mode DYNNORM)
- MD32431 $MA_MAX_AX_JERK [1] (maximum axial jerk for path motions in the dynamic response mode DYNPOS)

The machine data to be used is determined by the set positioning axis dynamic response mode:

MD18960 $MN_POS_DYN_MODE = <mode>

<mode>	Meaning
	The following is effective as maximum axial jerk for positioning axis motions:
0	MD32430 $MA_JOG_AND_POS_MAX_JERK With active G75 (fixed-point approach): MD32431 $MA_MAX_AX_JERK[0]
1	MD32431 $MA_MAX_AX_JERK[1]

10.5 Programming

10.5.1 General

Note

For the programming of position axes, please observe the following documentation:
References:
Programming Manual Basics; Section: "feed rate control" and "spindle motion"

Note

The maximum number of positioning axes that can be programmed in a block is limited to the maximum number of available channel axes.

Definition

Positioning axes are defined using the following parameters:

- Axis type: Positioning axis type 1, type 2 or type 3
- End point coordinates (in absolute dimensions or in incremental dimensions)
- Feedrate for linear axes in [mm/min], for rotary axes in [degrees/min]

Example: Positioning axis type 1

Program code	Comment
`POS[Q1]=200 FA[Q1]=1000`	`; Axis Q1 with feedrate 1000mm/min at Position 200.`

Example: Positioning axis type 2

Program code	Comment
POSA[Q2]=300 FA[Q2]=1500	; Axis Q2 with feedrate 1,500mm/min at Position 300.

Note

Within a part program, an axis can be a path axis or a positioning axis. Within a movement block, however, each axis must be assigned a unique axis type.

Programming in synchronized action

Axes can be positioned completely asynchronous to the part program from synchronized actions.

Example:

Program code	Comment
ID=1 WHENEVER $R==1 DO POS[Q4]=10 FA[Q3]=990	; The axial feedrate is specified permanently.

References:
Function Manual, Synchronized Actions

Block change

The block change can be adjusted for positioning axis types 1 and 2 with:

FINEA=<axis name> or
FINEA[<axis name>]

COARSEA=<axis name> or
COARSEA[<axis name>]

IPOENDA=<axis name> or
IPOENDA[<axis name>]

In Type 3 positioning axis, the block change within the brake ramp of the single interpolation can be set with:

IPOBRKA=<axis name> or
IPOBRKA(<axis name>[,<instant in time*>])

* Instant in time of the block change, referred to the braking ramp as a %

Absolute dimension / incremental dimension

The programming of the end point coordinates takes place in absolute dimension (`G90`) or in incremental dimension (`G91`).

Example	Meaning
	Programming the end point coordinates
`G90 POS[Q1]=200`	In absolute dimension
`G91 POS[Q1]=AC(200)`	In absolute dimension
`G91 POS[Q1]=200`	In incremental dimension
`G90 POS[Q1]=IC(200)`	In incremental dimension

Reprogram type 2 positioning axes

With type 2 positioning axes (motion across block limits), you need to be able to detect in the part program whether the positioning axis has reached its end position. Only then is it possible to reprogram this positioning axis (otherwise an alarm is issued).

If `POSA` and then `POSA` again with `IPOBRKA` (block change in the braking ramp) is programmed, an alarm is not issued. For more information, please refer to NC command `IPOBKA` in Section "Settable block change time".

Coordination (WAITP)

The coordination command `WAITP` enables you to designate a position in the NC program where the program is to wait until an axis programmed with `POSA` in a previous NC block has reached its end position.

`WAITP` exists in an internal block.

An explicit reference must be made to any axis for which the program is to wait.

Example:

Program code	**Comment**
`N10 G01 G90 X200 F1000 POSA[Q1]=200 FA[Q1]=500`	
`N15 X400`	
`N20 WAITP(Q1)`	`; The program processing is stopped automatically until Q1 is at position.`
`N25 X600 POS[Q1]=300`	`; Q1 is a positioning axis of Type 1 (feedrate FA[Q1] from block N10).`
`N30 X800 Q1=500`	`; Q1 is path axis (path feed F1000 from block N10).`

Tool offset

A tool length compensation for positioning axes can be implemented by means of an axial zero offset, allowing, for example, the positioning path of a loader to be altered. An example where the axial zero offset might be used in place of the tool length compensation is where a loader containing tools of various dimensions has to bypass an obstacle.

End of program

The program end (program status selected) is delayed until all axes (path axes + positioning axes) have reached their programmed end points.

10.5.2 Revolutional feed rate in external programming

The two following setting data can be used to specify that the revolutional feed rate of a positioning axis should be derived from another rotary axis/spindle:

SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE(revolutional feed rate for position axes/spindles)

SD42600 JOG_FEED_PER_REV_SOURCE (control of revolutional feed rate in JOG)

The following settings are possible:

Value	Description
0	No revolutional feed rate selected
>0	The revolutional feed rate is derived from the round axis/spindle with the machine axis index specified here.
-1	The revolutional feed rate is derived from the master spindle of the channel in which the axis/spindle is active.
-2	The revolutional feed rate is derived from the rotary axis/spindle with the machine axis index 0.
-3	The revolutional feed rate is derived from the master spindle of the channel in which the axis/spindle is active. No revolutional feed rate is selected if the master spindle is at a standstill.

10.6 Block change

Since path and positioning axes are interpolated separately, they reach their programmed end positions at different instants in time. If path and positioning axes are programmed in a block together, then the block change behavior depends on the programmable type of positioning axes.

Type 1: Block-related positioning axis

Properties:

- The block change is performed as soon as **all path and positioning axes** have reached their respective programmed end-of-motion criterion:
  - Path axes: `G601`, `G602`, `G603`
  - Positioning axes: `FINEA`, `COARSA`, `IPOENDA`
- Programming the positioning axis: `POS[<axis>]`

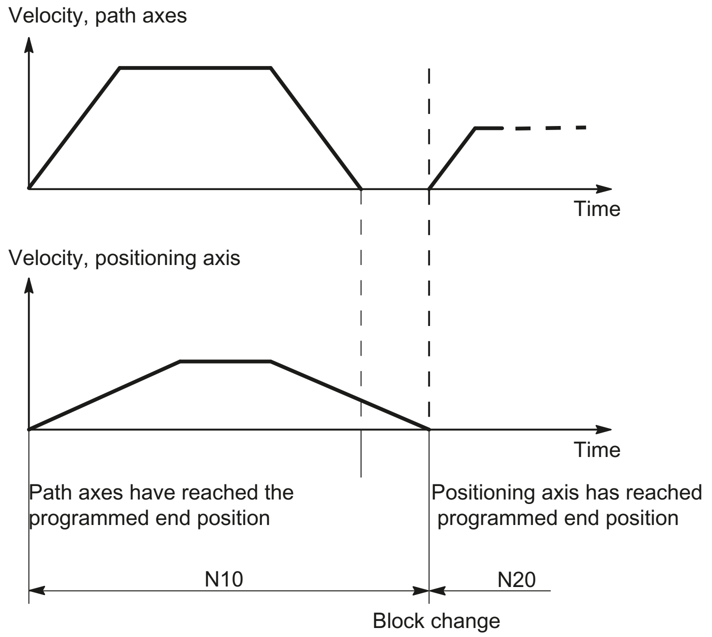


Figure 10-1 Block change for path axis and positioning axis type 1

Note

Continuous path mode

Continuous path mode across block limits (`G64`) is only possible if the positioning axes reach their end-of-motion criterion before the path axes (in the diagram above, this is not the case).

Type 2: Modal positioning axis (across blocks)

Properties:

- The block change is performed as soon as **all path axes** have reached their programmed end-of-motion criterion (`G601`, `G602`, `G603`)
- Programming the positioning axis: `POSA[<axis>]`
- The positioning axis traverses beyond the block limits to its programmed end position. It is not permissible that the positioning axis is programmed again before reaching its end-of-motion criterion.

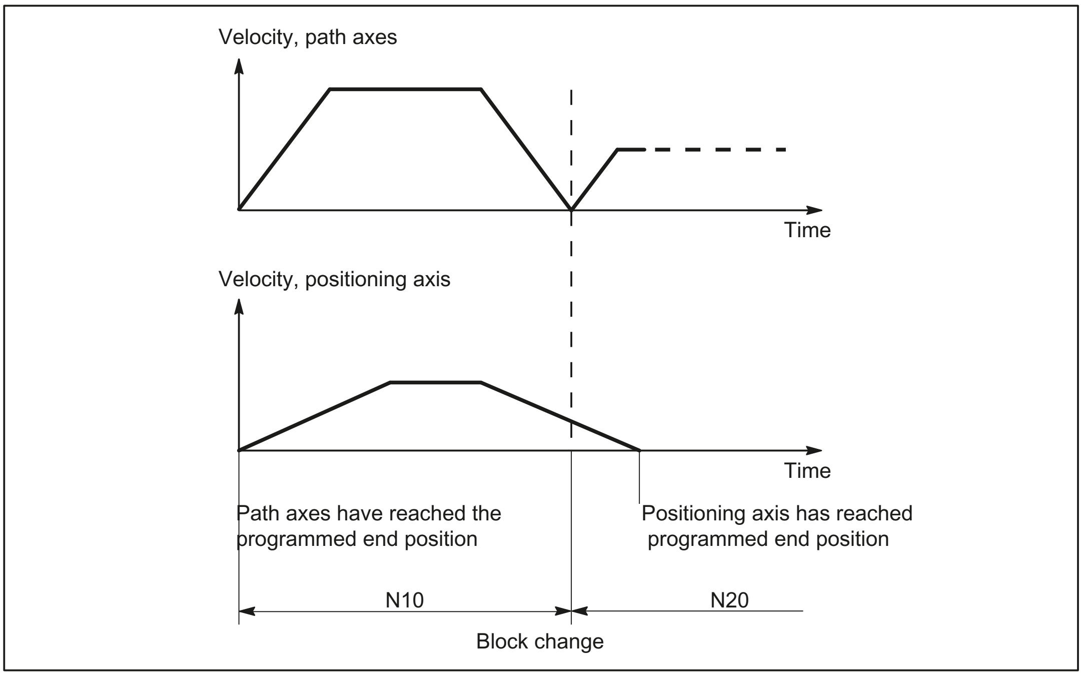


Figure 10-2 Block change for path axis and positioning axis type 2

10.6.1 Settable block change time

Type 3: Conditional block-related positioning axis

Properties:

- The block change is performed as soon as **all path and positioning axes** have reached their respective programmed end-of-motion criterion:
  - Path axes: `G601`, `G602`, `G603`
  - Positioning axes: `IPOBRKA`
- Programming the positioning axis:
  - `N(x) IPOBRK(<axis>[,<instant in time>]) ;own block`
  - `N(x+1) POS[<axis>]`

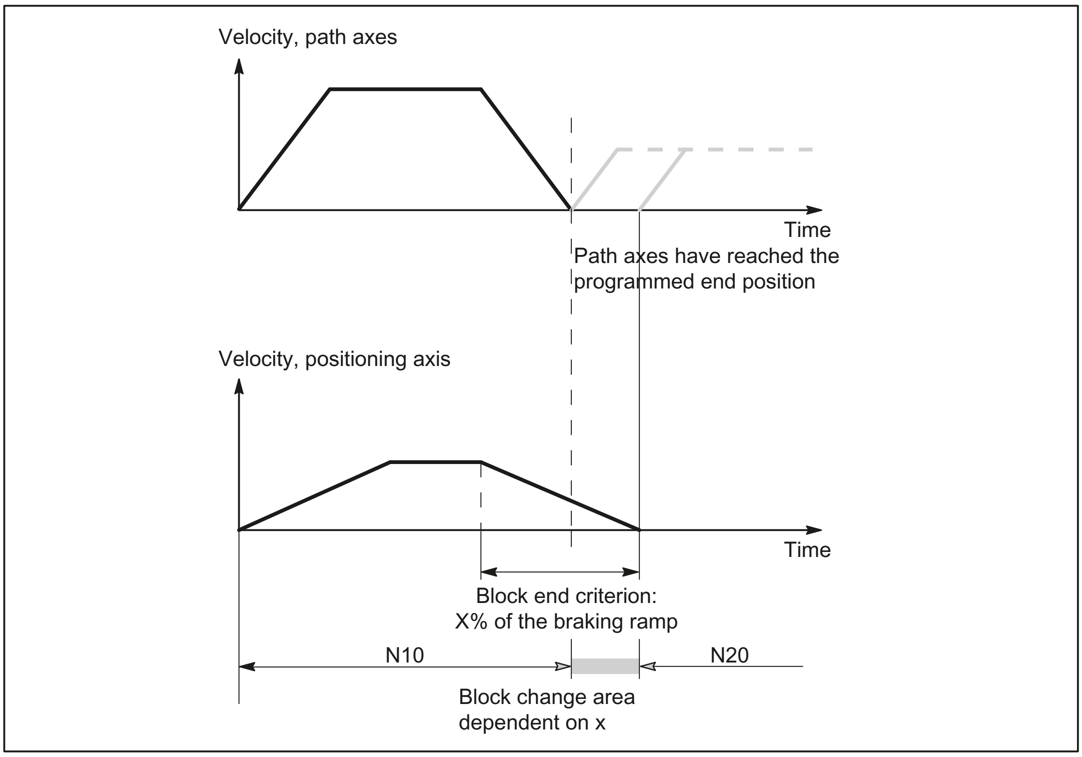


Figure 10-3 Block change for path axis and positioning axis type 3

Block change criterion: "Braking ramp" (IPOBRKA)

If, when activating the block change criterion "brake ramp", a value is programmed for the optional parameter <instant in time>, then this becomes effective for the next positioning motion and is written into the setting data synchronized to the main run. If no value is specified for the block change instant in time, then the actual value of the setting data is effective.

SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE

The time at which the block change can be realized is specified as a percentage of the braking ramp:

- 100% = start of the braking ramp
- 0% = end of the braking ramp, the same significance as block change criterion IPOENDA

Programming

```
IPOBRKA(<axis>[,<instant in time>])
```

IPOBRKA:	Block change criterion: Deceleration ramp Effective: Modal
<axis>:	Channel axis name (X, Y, ...)
<instant in time>:	Time of the block change, referred to the braking ramp as a %: • 100% = start of the braking ramp • 0% = end of the braking ramp, the same significance as IPOENDA Type: REAL

IPOBRKA is deactivated for the corresponding access when an axis end-of-motion criterion (FINEA, COARSEA, IPOENDA) is next programmed for the axis.

Additional block change criterion: "Tolerance window" (ADISPOSA)

A tolerance window around the end of block (either as actual or setpoint position) can be defined as additional block change criterion. Then, two conditions must be fulfilled for the block change:

- Block change criterion: "Braking ramp"
- Block change criterion: "Tolerance window"

Programming

```
ADISPOSA(<axis>[,<mode>,<window size>])
```

`ADISPOSA`:	Tolerance window for end-of-motion criterion Effective: Modal
`<axis>`:	Channel axis name (X, Y, ...)
`<mode>`:	Reference of the tolerance window Type: INT Value range: 0 Tolerance window not active 1 Tolerance window with respect to the setpoint position 2 Tolerance window with respect to actual position
`<window size>`:	Size of the tolerance window Type: REAL

System variable for end-of-motion criterion

The effective end-of-motion criterion can be read using the system variable $AA_MOTEND.

References: Parameter Manual, Book 2

Note

Information about other programmable end-of-motion criteria `FINEA`, `COARESA`, `IPOENDA` can be found in:

References: Function Manual, Basic Functions

- Spindles (S1), Section "Spindle modes"
- Feedrates (V1), Programmable dynamic response of single axis section

---

Supplementary conditions

Premature block change

A premature block change is not possible in the following cases:

- Oscillating axis

  During oscillation with partial infeed, the block-specific oscillation motion must remain active until the axis with partial infeed has reached its final position.
- Handwheel

  For handwheel input, the last set end-of-motion criterion applies.

Changing the axis state

The axis for which a block change occurred within the braking ramp can only be programmed in the following block in the same axis state.

When the axis state changes, e.g. to `POS`followed by `SPOS`, the last programmed end-of-motion criterion `FINEA`, `COARSEA`, `IPOENDA` is active. This also applies in the following cases:

- a positioning axis becomes a path axis
- if the program waits for the end of the positioning movement: `WAITP`, `M30`, end of the technology cycle, preprocessing stop
- Velocity override is deactivated or activated

Note

For further information about programming positioning axes, see:

References:

Programming Manual, Fundamentals, Section "Feedrate control and spindle motion"

Programming Manual, Advanced, Section "Special motion commands"

Examples

Block change criteria "braking ramp" in the part program

Program code	Comment
	; Default setting is effective.
N10 POS[X]=100	; Positioning motion from X to position 100.
	; Block change criterion: "Exact stop fine"
N20 IPOBRKA(X,100)	; Block change criterion: "Braking ramp", 100% = start of the braking ramp.
N30 POS[X]=200	; The block is changed as soon as the X axis starts to brake.
N40 POS[X]=250	; Axis X does not brake at position 200, but moves further to position 250. As soon as the axis starts to brake, the block is changed.
N50 POS[X]=0	; Axis X brakes and returns to position 0, the block change is realized at position 0 and "exact stop fine".
N60 X10 F100	; Axis X traverses as path axis to position 10.
N70 M30	

Block change criterion "braking ramp" in synchronized action

In the technology cycle:

Program code	Comment
FINEA	; End-of-motion criterion: "Exact stop fine"
N10 POS[X]=100	; The technology cycle block change is realized if the X axis has reached position 100 and "exact stop fine" is reached.
N20 IPOBRKA(X,100)	; Block change criterion, activate "braking ramp", 100% = start of the braking ramp.
N30 POS[X]=200	; The technology cycle block is changed as soon as the X axis starts to brake.
N40 POS[X]=250	; Axis X does not brake at position 200, but moves further to position 250. As soon as the axis starts to brake, the block change is realized in the technology cycle.
N50 POS[X]=0	; Axis X brakes and returns to position 0, the block change is realized at position 0 and "exact stop fine".
N60 M17	

Block change criterion "braking ramp" and "tolerance window" in the part program

Program code	Comment
	; Default setting is effective.
N10 POS[X]=100	; Positioning motion from X to position 100.
	; Block change criterion: "Exact stop fine"
N20 IPOBRKA(X,100)	; Block change criterion: "Braking ramp", 100% = start of the braking ramp.
N21 ADISPOSA(X,1,0.5)	; Tolerance window: 1 = setpoint position, tolerance = 0.5
N30 POS[X]=200	; The block is changed as soon as the X axis starts to brake.
N40 POS[X]=250	; Axis X does not brake at position 200, but moves further to position 250. As soon as the axis starts to brake, the block is changed.
N50 POS[X]=0	; Axis X brakes and returns to position 0, the block change is realized at position 0 and "exact stop fine".
N60 X10 F100	
N70 M30	

Block change criterion "braking ramp" and "tolerance window" in synchronized action

In the technology cycle:

Program code	Comment
FINEA	; End-of-motion criterion: "Exact stop fine"
N10 POS[X]=100	; The technology cycle block change is realized if the X axis has reached position 100 and "exact stop fine" is reached.
N20 IPOBRKA(X,100)	; Block change criterion, activate "braking ramp",100% = start of the braking ramp.
N21 ADISPOSA(X,2,0.3)	; Tolerance window: 2 = actual position, tolerance = 0.3
N30 POS[X]=200	; Technology cycle block change is realized as soon as the X axis starts to brake and the actual position of the X axis >= 199.7.
N40 POS[X]=250	; X axis does not brake at position 200, but moves further to position 250. As soon as the X axis starts to brake and the position of the X axis >= 249.7, the block change is realized in the technology cycle.
N50 POS[X]=0	; Axis X brakes and returns to position 0, the block change is realized at position 0 and "exact stop fine".
N60 M17	

`IPOBRKA(X)` could also be written into the `N20` blocks without specifying the instant in time, if the corresponding value has already been entered into the setting data:

SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE[X] == 100

See also

Control by the PLC (Page 629)

10.6.2 End of motion criterion with block search

Last block serves as container

The last end-of-motion criterion programmed for an axis is collected and output in an action block. The last block with a programmed motion end condition that was processed in the search run serves as a container for setting all axes.

Example

For two action blocks with end-of-motion criteria for three axes:

Program code	Comment
N01 G01 POS[X]=20 POS[Y]=30	
IPOENDA[X]	; Last programmed IPOENDA end-of-motion criterion
N02 IPOBRKA(Y, 50)	; Second action block for the Y axis IPOENDA
N03 POS[Z]=55 FINEA[Z]	; Second action block for the Z axis FINEA
N04 $A_OUT[1]=1	; First action block for output as digital output
N05 POS[X]=100	;
N06 IPOBRKA(X, 100)	; Second action block for the X axis IPOBRKA container
...	; For all programmed end-of-motion criteria
TARGET:	; Block search target

The first action block contains the digital output:

$A_OUT[1] = 1.

The second action block contains the settings for the end-of-motion criteria:

for the X axis IPOBRKA / $SA_IPOBRAKE_BLOCK_EXCHANGE[AX1]=100

for the Y axis IPOBRKA / $SA_IPOBRAKE_BLOCK_EXCHANGE[AX2]=50

for the Z axis FINEA. The motion end condition IPOENDA last programmed is also stored for the X axis.

10.7 Control by the PLC

PLC axes

PLC axes are traversed from the PLC and can move asynchronously to all other axes. The travel motions are executed separate from the path and synchronized actions.

Reference:
Function Manual, Basic Functions; Basic PLC Program for SINUMERIK 840D sl (P3) or PLC for SINUMERIK 828D (P4)

Concurrent positioning axes

Using function block FC18, for SINUMERIK 840D sl concurrent positioning axes can be started from the PLC.

Channel-specific signals

All channel-specific signals act to the same extent on path and positioning axes.

Only the following signals are an exception:

- IS DB21, ... DBB4 ("Feedrate override")
- IS DB21, ... DBX6.2 ("Delete distance to go")

Axisspecific signals

Positioning axes have the following additional signals:

- IS DB31, ... DBX76.5 ("Positioning axis")
- Feedrate for positioning axes/spindles (FA)
- IS DB31, ... DBB0 ("Feedrate override") axis-specific
- IS DB31, ... DBX2.2 ("Distance to go/Spindle reset") Axis-specific deletion of distance to go

Single-axis functions of PLC-controlled axes

The behavior of individual PLC axes can be manipulated in the following way with machine data:
MD30460 $MA_BASE_FUNCTION_MASK
:

- Bit 4 = 1

  The axis is exclusively controlled by the PLC.
- Bit 5 = 1

  The axis is a permanently assigned PLC axis.
- Bit 6 = 1

  The channel-specific NC/PLC interface signal:
  DB21, ... DBX6.0 ("feed disable")
  is not active for the axis if this is a PLC-controlled axis.
- Bit 7= 1

  The channel-specific NC/PLC interface signal:
  DB21, ... DBX36.3 ("all axes stationary")
  is set**independently** of an axis if this is a PLC-controlled axis.

For a PLC-controlled axis:

- The channel-specific NC/PLC interface signal DB21, ... DBX6.0 ("feed disable"), is active if bit 6 = 0 in machine data MD30460 $MA_BASE_FUNCTION_MASK.
- The channel-specific NC/PLC interface signal DB21, ... DBX36.3 ("all axes stationary") is only set if bit 7 = 0 in machine data MD30460 $MA_BASE_FUNCTION_MASK.

If an attempt is made to assign an **exclusively PLC-controlled axis** to the NC program or to request the axis for the NC program, then this is rejected with Alarm 26075. Alarm 26076 is generated in the same way for a PLC axis with fixed assignment.

A **PLC axis with fixed assignment** is a "neutral axis" on power up. For a travel request via the NC/PLC interface, a concurrent positioning axis automatically changes to a PLC axis without being interchanged beforehand.

Axis replacement via PLC

The type of an axis for axis replacement is transferred to the PLC with axial interface byte NCK→PLC NST DB31, ... DBB68 (see alsop Section "K5: Mode groups, channels, axis interchange (Page 305)"):

- IS DB31, ... DBX68.0-68.3 ("NC axis/spindle in channel") channels 1 to 10
- IS DB31, ... DBX68.4 ("new type requested by PLC")
- IS DB31, ... DBX68.5 ("axis can be replaced")
- IS DB31, ... DBX68.6 ("neutral axis/spindle")
- IS DB31, ... DBX68.7 ("PLC axis/spindle")

If IS DB31, ... DBX68.5 ("axis can be replaced") = 1, an axis replacement request can be issued from the PLC.

10.7.1 Starting concurrent positioning axes from the PLC

Activation from PLC

When concurrent positioning axes are activated from the PLC, FC18 is called and supplied with the following parameter data:

- Axis name or axis number
- End position
- Feedrate (with feedrate setting = 0, the feedrate is determined by the setting in machine data MD32060 $MA_POS_AX_VELO):
- Absolute coordinate (G90),
  incremental coordinate (G91),
  absolute coordinate along the shortest path for rotary axes (rotary axis name = DC(value))

The following functions are defined:

- Linear interpolation (G01)
- Feedrate in mm/min or degrees/min (G94)
- Exact stop (G09)
- Settable zero offsets currently selected are valid

Since each axis is assigned to exactly one channel, the control can select the correct channel from the axis name/axis number and start the concurrent positioning axis on this channel.

10.7.2 PLC-controlled axes

PLC actions

The table below compares the following PLC actions with the corresponding NCK reactions for a machine axis 1:

- Start machine axis as PLC axis via FC 18
- Initiate `NC start` or `NC stop`
- Trigger axial `STOP`, `RESUME` or `RESET`
- Trigger `NC-RESET`
- Cancel or set controller enable for the machine axis
- Relinquish control of machine axis to NC

Examples of NCK reactions

PLC actions are shown as NCK reactions in the table below.

PLC actions	NCK reaction
Start machine AX1, residing in the 1st channel, as PLC axis via FC 18	
Initiate NC stop axes and spindles DB21, ... DBX7.4 = 1 (NC stop axes plus spindle)	AX1 is stopped.
DB21, ... DBX7.1 = 1 initiate (NC start)	AX1 continues to traverse.
PLC wants to control AX1, DB31, ... DBX28.7 = 1 (PLC controls axis)	Checking AX1 is relinquished to the PLC. DB31, ... DBX63.1 = 1 (PLC controls the axis)
Initiate NC stop for axes and spindles DB21, ... DBX7.4 = 1 ("NC stop axes plus spindle")	AX1 is not stopped.
Initiate axial stop DB31, ... DBX28.6 = 1 (stop along braking ramp)	AX1 is stopped DB31, ... DBX63.2 ==1 (axis stop active)
Initiate that axial movement continues DB31, ... DBX28.2 = 1 (continue)	AX1 moves further DB31, ... DBX63.2 == 0 (axis stop active)
Initiate NC-RESET DB21, ... DBX7.7 = 1 initiate (reset)	No effect on AX1
Initiate axial reset DB31, ... DBX28.1 = 1 (reset)	AX1 is stopped and the traversing motion is interrupted: • DB31, ... DBX63.2 = 0 (axis stop active) • Read-in axial machine data • DB31, ... DBX63.0 = 1 (reset executed) • DB31, ... DBX63.2 = 0 (axis stop active)

PLC actions	NCK reaction
Start machine axis AX1 as PLC axis via FC 18	DB31, ... DBX63.0 = 0 (reset executed)
Withdraw controller enable for AX1: DB31, ... DBX2.1 = 0 (controller enable)	Alarm 21612 "Axis %1 measuring system change" is displayed
Initiate that axial movement continues DB31, ... DBX28.2 = 1 (continue)	• Alarm 21612 "Axis %1 measuring system change" is deleted • DB21, ... DBX40.7 = 1(traversing command plus) • AX1 does not traverse due to a missing controller enable signal.
Set controller enable for AX1 DB31, ... DBX2.1 = 1 (controller enable)	AX1 moves to the programmed end point.
Initiate axial reset DB31, ... DBX28.1 = 1 (reset)	• Stop AX1 • Read-in axial machine data • DB31, ... DBX63.0 = 0 (reset)
PLC relinquishes control of AX1 to the NCK from DB31, ... DBX28.7 = 0 (PLC controls axis)	• NCK accepts control of machine axis • DB31, ... DBX63.1 = 0 (PLC controls the axis) • DB31, ... DBX63.0 = 0 (reset)

10.7.3 Control response of PLC-controlled axes

Response to channel reset, NEWCONFIG, block search and MD30460

Control response to	PLC-controlled axis
Mode change and NC program control	work independently of axis.
Channel RESET	No axial machine data are effective and a traversing movement is not aborted.
NEWCONFIG	No axial machine data are effective.
Block search Type 5 SERUPRO	are processed during SERUPRO to simulate the normal procedure, e.g. PLC takes over or relinquishes control of this axis which is also traversing via the PLC.
All block search variants of types 1, 2, 4 and 5	The PLC takes over control of the axis **before** the approach block and is responsible for positioning this axis.
NC-controlled retraction activated with $AC_ESR_TRIGGER.	has no effect and acts only on the specified PLC-controlled axis.

Control response to	PLC-controlled axis
machine data: MD30460 $MA_BASE_FUNKTION_MASK	**which is not controlled exclusively by the PLC**
Bit 4 = 0	**cannot** be changed directly with axis replacement command GET (axis) or AXTOCHAN(axis, channel) to an axis controlled by the NC program, see *** Note on axis replacement**.
Bit 4 = 1	cannot be requested for the NC program. GET or AXTOCHAN from the NC program or a synchronized action, or programming the axis in the NC program, are rejected with alarm 26075.
MD30460 $MA_BASE_FUNKTION_MASK	**For the PLC-controlled axis**
Bit 6 = 1	channel-specific IS DB 21, ... DBX6.0 ("feed disable") is not effective. This axis is not stopped when feed disable is activated, but continues to move.
Bit 7 = 1	the axis is not taken into account when IS DB 21, ... DBX36.3 ("all axes stationary") is generated. This signal is output with 1 even if all other axes in the channel are stationary and only the PLC-controlled axis is active.

*** Note on axis replacement**

This replacement of a "neutral axis" by an "NC program axis" does not take place until the PLC has really relinquished control of the axis in accordance with use case "Relinquish control of axis". Waiting for this axis interchange is displayed on the HMI operator panel front.

10.8 Response with special functions

10.8.1 Dry run (DRY RUN)

The dry run feedrate is also effective for positioning axes unless the programmed feedrate is larger than the dry run feedrate.

Activation of the dry run feed entered in SD42100 $SA_DRY_RUN_FEED can be controlled with SD42101 $SA_DRY_RUN_FEED_MODE. See

References:

Function Manual, Basic Functions; Feedrates (V1)

10.8.2 Single block

Positioning axis type 1

Single-block mode is effective with positioning axes of type 1.

Positioning axis type 2

Positioning axes of type 2 also continue across block limits in single block mode.

Positioning axis type 3

Positioning axes of type 3 also continue across block limits in single block mode.

10.9 Examples

10.9.1 Motion behavior and interpolation functions

In the following example, the two positioning axes Q1 and Q2 represent two separate units of movement. There is no interpolation relationship between the two axes. In the example, the positioning axes are programmed as type 1 (e.g. in N20) and type 2 (e.g. in N40).

Program example

Program code

```
N10 G90 G01 G40 T0 D0 M3 S1000
N20 X100 F1000 POS[Q1]=200 POS[Q2]=50 FA[Q1]=500
FA[Q2]=2000
N30 POS[Q2]=80
N40 X200 POSA[Q1] = 300 POSA[Q2]=200] FA[Q1]=1500
N45 WAITP[Q2]
N50 X300 POSA[Q2]=300
N55 WAITP[Q1]
N60 POS[Q1]=350
N70 X400
N75 WAITP[Q1, Q2]
N80 G91 X100 POS[Q1]=150 POS[Q2]=80
N90 M30
```

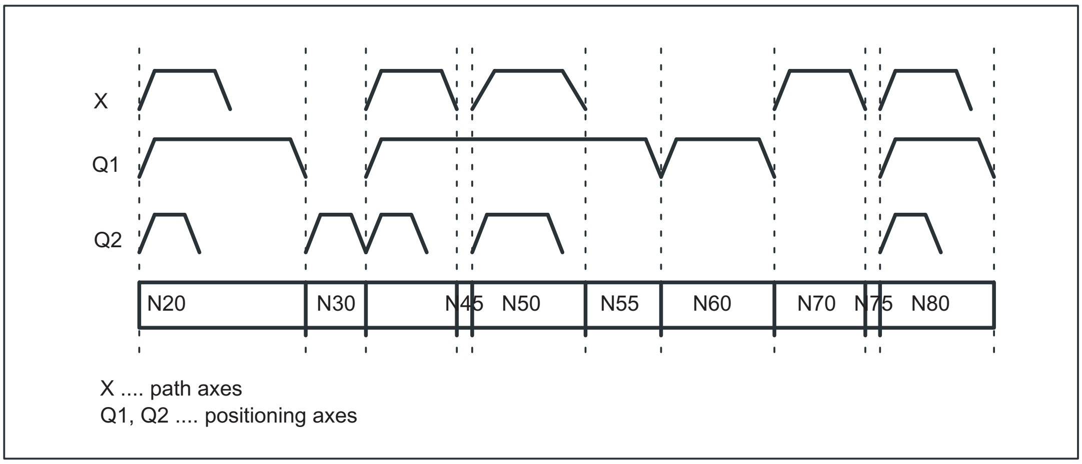


Figure 10-4 Timing of path axes and positioning axes

10.9.1.1 Traversing path axes without interpolation with G0

Example in G0 for positioning axes

Path axes traverse as positioning axes with no interpolation in rapid traverse mode (G0):

Program code	Comment
	; Activation of nonlinear
	; Interpolation
	; MD20730 $MC_GO_LINEAR_MODE == FALSE
	; is set
G0 X0 Y10	; axis traverses without interpolation
G0 G43 X20 Y20	; axis traverses in path mode (with interpolation)
	; Traversing
G0 G64 X30 Y30	; axis traverses in path mode (with interpolation)
	; Traversing
G0 G95 X100 Z100 m3 s100	; axis traverses without interpolation
	; no revolutional feedrate active

10.10 Data lists

10.10.1 Machine data

10.10.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
18960	POS_DYN_MODE	Type of positioning axis dynamic response

10.10.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20730	G0_LINEAR_MODE	Interpolation behavior with G0
20732	EXTERN_G0_LINEAR_MODE	Interpolation behavior with G00
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions

10.10.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30450	IS_CONCURRENT_POS_AX	Concurrent positioning axis
30460	BASE_FUNCTION_MASK	Axis functions
32060	POS_AX_VELO	Feedrate for positioning axis
32300	MAX_AX_ACCEL	Maximum axis acceleration
32430	JOG_AND_POS_MAX_JERK	Maximum axial jerk for positioning axis movements
32431	MAX_AX_JERK	Maximum axial jerk for path movements
37510	AX_ESR_DELAY_TIME1	Delay time for ESR single axis
37511	AX_ESR_DELAY_TIME2	ESR time for interpolatory braking of single axis

10.10.2 Setting data

10.10.2.1 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43600	IPOBRAKE_BLOCK_EXCHANGE	Braking ramp block change condition
43610	ADISPOSA_VALUE	Braking ramp tolerance window

10.10.3 Signals

10.10.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
feed disable	DB21, ...DBX6.0	DB3200.DBX6.0
NC Start	DB21, ...DBX7.1	DB3200.DBX7.1
NC stop axes plus spindle	DB21, ...DBX7.4	DB3200.DBX7.4
Reset	DB21, ...DBX7.7	-

10.10.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
All axes stationary	DB21, ...DBX36.3	DB3300.DBX4.3
Travel command minus	DB21, ...DBX40.6	DB3300.DBX1000.6
Travel command plus	DB21, ...DBX40.7	DB3300.DBX1000.7

10.10.3.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate override, axis-specific	DB31, ...DBB0	DB380x.DBB0
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Delete distance-to-go spindle reset for specific axes	DB31, ...DBX2.2	DB380x.DBX2.2
Reset	DB31, ...DBX28.1	-
Continue	DB31, ...DBX28.2	-
Stop along braking ramp	DB31, ...DBX28.6	-
PLC-controlled axis	DB31, ...DBX28.7	-

10.10.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Exact stop coarse	DB31, ...DBX60.6	DB390x.DBX0.6
Exact stop fine	DB31, ...DBX60.7	DB390x.DBX0.7
Axial alarm	DB31, ...DBX61.1	DB390x.DBX1.1
Axis ready (AX_IS_READY)	DB31, ...DBX61.2	DB390x.DBX1.2
Axis container rotation active	DB31, ...DBX62.7	-
AXRESET DONE	DB31, ...DBX63.0	-
PLC-controlled axis	DB31, ...DBX63.1	-
Axis stop active	DB31, ...DBX63.2	DB390x.DBX3.2
Travel command minus	DB31, ...DBX64.6	DB390x.DBX4.6
Travel command plus	DB31, ...DBX64.7	DB390x.DBX4.7
Positioning axis	DB31, ...DBX76.5	DB390x.DBX1002.5
F function (feedrate) for positioning axis	DB31, ...DBB78-81	-
Emergency retraction active	DB31, ...DBX98.7	DB390x.DBX5002.7

P5: Oscillation - only 840D sl 11

11.1 Brief description

Definition

When the "Oscillation" function is selected, an oscillation axis oscillates backwards and forwards at the programmed feedrate or a derived feedrate (revolutional feedrate) between two reversal points. Several oscillation axes can be active at the same time.

Oscillation variants

Oscillation functions can be classified according to the axis response at reversal points and with respect to infeed:

- Asynchronous oscillation across block boundaries

  During reciprocating movement, other axes can interpolate at will. The oscillation axis can act as the input axis for dynamic transformation or as the master axis for gantry or coupled-motion axes. Oscillation is not automatically linked to the AUTOMATIC mode.
- Oscillation with continuous infeed.

  Simultaneous infeed in multiple axes is possible. However, there is no interpolative connection between the infeed and oscillation movements.
- Oscillation with infeed in both reversal points or only in the left-hand or right-hand reversal point. The infeed can be initiated at a programmable distance from the reversal point.
- Sparking-out strokes after oscillation.
- Beginning and end of oscillation at defined positions.

Response at reversal points

The change in direction is initiated:

- without the exact stop limit being reached (exact stop fine or coarse)
- After reaching the programmed position or
- after the programmed position is reached and expiry of a dwell.
- by an external signal (from the PLC).

Control methods

Oscillation movements can be controlled by various methods:

- The oscillation movement and/or infeed can be interrupted by delete distance-to-go.
- The reversal points can be altered via NC program, PLC, HMI, handwheel or directional keys.
- The feedrate velocity of the oscillation axis can be altered through a value input in the NC program, PLC, HMI or via an override. The feedrate can be programmed to be dependent on a master spindle, rotary axis or spindle (revolutional feedrate).

  References:

  Function Manual, Basic Functions; Feedrates (V1)
- The oscillation movement can be controlled entirely by the PLC.

Methods of oscillation control

There are two modes of oscillation:

1. Asynchronous oscillation:

   Is active across block boundaries and can also be started from the PLC/HMI.
2. Oscillation by synchronized movement actions:

   In this case the asynchronous oscillation and an infeed movement are coupled via synchronized actions. In this way, it is possible to program oscillation with infeed at the reversal points which is active on a non-modal basis.

11.2 Asynchronous oscillation

Characteristics

The characteristics of asynchronous oscillation are as follows:

- The oscillation axis oscillates backwards and forwards between reversal points at the specified feedrate until the oscillation movement is deactivated or until there is an appropriate response to a supplementary condition. If the oscillation axis is not positioned at reversal point 1 when the movement is started, then it traverses to this point first.
- Linear interpolation G01 is active for the oscillation axis regardless of the G code currently valid in the program. Alternately, revolutional feedrate G95 can be activated.
- Asynchronous oscillation is active on an axis-specific basis beyond block limits.
- Several oscillation axes (i.e. maximum number of positioning axes) can be active at the same time.

- During the oscillation movement, axes other than the oscillation axis can be freely interpolated. A continuous infeed can be achieved via a path movement or with a positioning axis. In this case, however, there is no interpolative connection between the oscillation and infeed movements.
- If the PLC does not have control over the axis, then the axis is treated like a normal positioning axis during asynchronous oscillation. In the case of PLC control, the PLC program must ensure via the appropriate stop bits of the NC/PLC interface that the axis reacts in the desired way to NC/PLC interface signals. These NC/PLC interface signals also include end of program, operating mode change and single block.
- The oscillation axis can act as the input axis for the transformations (e.g. inclined axis, see Section "M1: Kinematic transformation (Page 347)").
- The oscillation axis can act as the master axis for gantry and coupled motion axes.

  References:

  Function Manual, Special Functions; Gantry Axes (G1)
- It is possible to traverse the axis with jerk limitation (SOFT) and/or with kneeshaped acceleration characteristic (as for positioning axes).
- In addition to this, the oscillation movement can be activated in synchronism with the block via the part program.
- The oscillation movement can likewise be started, influenced and stopped from the PLC/HMI.
- Interpolatory oscillation is not possible (e.g. oblique oscillation).

11.2.1 Influences on asynchronous oscillation

Setting data

The setting data required for oscillation can be set with special language commands in the NCK part program, via the HMI and/or the PLC.

Feedrate

The feed velocity for the oscillation axis is selected or programmed as follows:

- The velocity defined for the axis as a positioning axis is used as the feed velocity. This value can be programmed via FA[axis] and has a modal action. If no velocity is programmed, then the value stored in machine data POS_AX_VELO is used (see also Section "P2: Positioning axes (Page 599)").
- When an oscillation movement is in progress, the feed velocity of the oscillation axis can be altered via setting data. It can be specified via the part program and setting data whether the changed velocity must take effect immediately or whether it should be activated at the next reversal point.

- The feedrate can be influenced by the override (axial NC/PLC interface signal and programmable).
- If Dry Run is active, the dry run velocity setting is applied if it is higher than the currently programmed velocity.

  Activation of the dry run feed entered in SD42100 $SC_DRY_RUN_FEED can be controlled with SD42101 $SC_DRY_RUN_FEED_MODE.

  References:

  Function Manual, Basic Functions; Feedrates (V1)
- The velocity overlay / path overlay can be influenced by the handwheel (see following table and Section "H1: Manual and handwheel travel (Page 143)").
- The oscillation axis can be moved with reversal feed.

Reversal feed

The reversal feed can also be used for oscillation axes.

Reversal points

The positions of the reversal points can be entered via setting data before an oscillation movement is started or while one is in progress.

- The reversal point positions can be entered by means of manual travel (handwheel, JOG keys) before or in the course of an oscillation movement, regardless of whether the oscillation movement has been interrupted or not.

The following applies to alteration of a reversal point position: When an oscillation movement is already in progress, the altered position of a reversal point does not become effective until this point is approached again. If the axis is already approaching the position, the correction will take effect in the next oscillation stroke.

Note

If a reversal point must be altered at the same time as NC/PLC interface signal DB21, … DBX0.3 ("Activate DRF") is set, the handwheel signals are applied both to the DRF offset and to the offset of the reversal point, i.e. the reversal point is shifted absolutely by an amount corresponding to twice the distance.

Stop times

A stop time can be programmed via setting data for every reversal point.

The setting can be changed in the following blocks of the NC program. It is then effectively block synchronous from the next reversal point.

Stop time can be changed asynchronously via setting data. It is then effective from the instant that the appropriate reversal point is next traversed.

The following table explains the motional behavior in the exact stop range or at the reversal point, depending on the stop time input.

Table 11- 1 Effect of stop time

Stop time setting	Response
-2	Interpolation continues without wait for exact stop
-1	Wait for coarse exact stop at reversal point
0	Wait for fine exact stop at reversal point
>0	Wait for fine exact stop at reversal point, followed by wait for stop time.

Deactivate oscillation

One of the following options can be set for termination of the oscillation movement when oscillation mode is deactivated:

- Termination of oscillation movement at the next reversal point
- Termination of oscillation movement at reversal point 1
- Termination of oscillation movement at reversal point 2

Following this termination process, sparking-out strokes are processed and an end position approached if programmed.

On switchover from asynchronous oscillation to spark-out and during spark-out, the response at the reversal point regarding exact stop corresponds to the response determined by the stop time programmed for the appropriate reversal point. A sparking-out stroke is the motion to the other reversal point and back (see the table below):

Note

Oscillation with motion-synchronous actions and stop times "OST1/OST2"

Once the set stop times have expired, the internal block change is executed during oscillation (indicated by the new distances-to-go of the axes). The deactivation function is checked when the block changes. The deactivation function is defined according to the control setting for the motional sequence "OSCTRL".

This dynamic response can be influenced by the feed override.

An oscillation stroke may then be executed before the sparking-out strokes are started or the end position approached.

Although it appears as if the deactivation response has changed, this is not in fact the case.

Table 11- 2 Operational sequence for deactivation of oscillation

Function	Inputs	Explanation
Deactivation at defined reversal point	Number of sparking-out strokes equals 0, no end position active	The oscillation movement is stopped at the appropriate reversal point
Deactivation with specific number of sparking-out strokes	Number of sparking-out strokes is not equal to 0, no end position is active	After the appropriate reversal point is reached, the number of sparking-out strokes specified in the command are processed.
Deactivation with sparking-out strokes and defined end position (optional)	Number of sparking-out strokes is not equal 0 end position active	After the appropriate reversal point is reached, the number of sparking-out strokes specified in the command are processed, followed by approach to specified end position.
Deactivation without sparking-out strokes, but with defined end position (optional)	Number of sparking-out strokes is equal 0 end position active	After the appropriate reversal point is reached, the axis is traversed to the specified end position.

NC language

The NC programming language allows asynchronous oscillation to be controlled from the part program. The following functions allow asynchronous oscillation to be activated and controlled as a function of NC program execution.

Note

If the setting data is directly written in the part program, then the data change takes effect prematurely with respect to processing of the part program (at the preprocessing time). It is possible to re-synchronize the part program and the oscillation function commands by means of a preprocessing stop (`STOPRE`).

References:

Programming Guide

1) Activate, deactivate oscillation:

- OS[oscillation axis] = 1; Activate oscillation for oscillation axis
- OS[oscillation axis] = 0; Deactivate oscillation for oscillation axis

---

Note

Every axis may be used as an oscillation axis.

---

2) End of oscillation:

- WAITP(oscillation axis)

  Positioning axis command – stops block until oscillation axis is at fine stop and synchronizes preprocessing and main run. The oscillation axis is entered as positioning axis again and can then be used normally.

  If an axis is to be used for oscillation, it must be released with a WAITP(axis) command beforehand.

  This also applies if oscillation is initiated from the PLC/HMI. In this case, the WAITP(axis) call is also needed if the axis was programmed beforehand via the NC program. As of SW version 3.2 it is possible to select via machine data $MA_AUTO_GET_TYPE, whether WAITP() shall be performed with programming or automatically.

Note

WAITP effectively implements a time delay until the oscillation movement has been executed. Termination of the movement can be initiated, for example, through a programmed deactivation command in the NC program or via the PLC or HMI by means of deletion of distance-to-go.

---

3) Setting reversal points:

- OSP1[oscillation axis] = position of reversal point 1
- OSP2[oscillation axis] = position of reversal point 2

A position is entered into the appropriate setting data in synchronism with the block in the main run and thus remains effective until the setting data is next changed.

If incremental traversal is active, then the position is calculated incrementally to the last appropriate reversal point programmed in the NC program.

4) Stopping times at reversal points:

- OST1[oscillation axis] = stop time at reversal point 1 in [s]
- OST2[oscillation axis] = stop time at reversal point 2 in [s]

A stop time is entered into the appropriate setting data in synchronism with the block in the main run and thus remains effective until the setting data is next changed.

The unit for the stop time is identical to the unit selected for the stop time programmed with G04.

5) Setting feedrate:

- FA[axis] = FValue

  Positioning axis infeed.

The feedrate is transferred to the appropriate setting data in synchronism with the block in the main run. If the oscillation axis is moved with reversal feed, the corresponding dependencies must be indicated as described in Description of Functions V1.

6) Setting control settings for sequence of movements:

- OSCTRL[oscillating axis] = (set options, reset options)

  The set options are defined as follows (the reset options deselect the settings):

Table 11- 3 Set/reset options

Option value	Meaning
0	Stop at next reversal point on deactivation of the oscillation movement (default). Can only be achieved by resetting option values 1 and 2.
1	Stop at reversal point 1 on deactivation of the oscillation movement
2	Stop at reversal point 2 on deactivation of the oscillation movement
3	On deactivation of oscillation movement, do not approach reversal point unless sparking-out strokes are programmed
4	Approach an end position after spark-out process
8	If the oscillation movement is aborted by delete distance-to-go, the sparking-out strokes must then be executed and the end position approached (if programmed)
16	If the oscillation movement is terminated by deletion of distance-to-go, the programmed reversal point must be approached on deactivation of the oscillation movement.
32	Altered feedrate will only take effect from the next reversal point.
64	If feedrate setting is 0, path overlay is active, or otherwise velocity overlay
128	For rotary axis DC (shortest path)
256	Sparking-out stroke as single stroke
512	First approach start position

Note

The option values 0-3 encode the behavior at reversal points at Power OFF. You can choose one of the alternatives 0-3. The other settings can be combined with the selected alternative according to individual requirements. A + character can be inserted to create a string of options.

Example: The oscillation movement for axis Z must stop at reversal point 1 on deactivation; an end position must then be approached and a newly programmed feedrate take immediate effect; the axis must stop immediately after deletion of distance-to-go.

OSCTRL[Z] = (1+4, 16+32+64)

The set/reset options are entered into the appropriate setting data in synchronism with the block in the main run and thus remain effective until the setting data is next changed.

Note

The control evaluates the reset options, then the set options.

---

7) Sparking-out strokes:

- OSNSC[oscillation axis] = number of sparking-out strokes

The number of sparking-out strokes is entered into the appropriate setting data in synchronism with the block in the main run and thus remains effective until the setting data is next changed.

8) End position to be approached after deactivation of oscillation:

- OSE[oscillation axis] = end position of oscillation axis

The end position is entered into the appropriate setting data in synchronism with the block in the main run and thus remains effective until the setting data is next changed. Option value 4 is implicitly set so that the set end position is approached.

9) Start position to be approached prior to activation of oscillation:

- OSB [oscillation axis] = start position of oscillation axis

The start position is entered into the appropriate setting data SD43790 $SA_OSCILL_START_POS in synchronism with the block in the main run and thus remains effective until the setting data is next changed. Bit 9 in setting data SD43770 $SA_OSCILL_CTRL_MASK must be set to initiate an approach to the start position. The start position is approached **before reversal point 1**. If the start position coincides with reversal position 1, reversal position 2 is approached next.

As an alternative to programming command OSB, it is also possible to enter the start position directly in setting data SD43790 $SA_OSCILL_START_POS.

All positional information in the setting data and system variables refer to the basic coordinate system (BCS). The positional data for OSB, OSE refer to the workpiece coordinate system (WCS).

No halt time applies when the start position is reached, even if this position coincides with reversal position 1; instead, the axis waits for the exact stop fine signal. Any configured exact stop condition is fulfilled.

If a non-modal oscillation process does not require an infeed motion if the start position coincides with reversal position 1, this option can be configured with another synchronized action (see Section "Non-modal oscillation (starting position = reversal point 1) (Page 673)").

Programming example

An example that contains all the important elements for asynchronous oscillation can be found in Section "Example of asynchronous oscillation (Page 666)".

11.2.2 Asynchronous oscillation under PLC control

Activation

The function can be selected from the PLC using the following setting data in all operating modes except for MDA Ref and JOG Ref.:

SD43780 OSCILL_IS_ACTIVE (switch-on oscillation motion)

Settings

The following criteria can be controlled from the PLC via setting data: Activation and deactivation of oscillation movement, positions of reversal points, stop times at reversal points, feedrate velocity, the options in the reversal points, the number of sparking-out strokes and the end position after deactivation. However, these values can also be set beforehand as a setting data via the HMI user interface directly or via an NC program. These settings remain valid after power ON and the PLC can also start an oscillation movement set in this way directly via setting data OSCILL_IS_ACTIVE (via variable service).

Supplementary conditions

A spindle which must act as an axis to execute an oscillation movement started via the PLC must fulfill the conditions required to allow traversal as a positioning axis, i.e. the spindle must, for example, have been switched to the position control (`SPOS`) beforehand.

The axes always respond to the following two stop bits - regardless of whether the axis is controlled from the PLC or not:

- DB31, ... DBX28.5 (stop at the next reversal point)
- DB31, ... DBX28.6 (stop along braking ramp)

11.2.3 Special reactions during asynchronous oscillation

With PLC control

The PLC program can assume the control of an oscillation axis via NC/PLC signals. The NC/PLC interface signals also include end of program, operating mode change and single block.

Using the NCU system software, an asynchronous reciprocating axis interpolated by the main run reacts to `NC STOP`, alarm handling, end of program, program control and `RESET`.

The PLC controls the axis/spindle via the axial NC/PLC interface (PLC→NCK) by means of IS DB31, ... DBX28.7 (PLC controlled axis) = 1

For further information about axes with PLC control, see Section "P2: Positioning axes (Page 599)".

Without PLC control

If the PLC does not have control over the axis, then the axis is treated like a normal positioning axis (POSA) during asynchronous oscillation.

Delete distance-to-go

Channel-specific delete distance-to-go is ignored.

Axial delete distance-to-go

- Without PLC control: Stop via braking ramp
- With PLC control: No stop (must be initiated from the PLC)

The following applies to **both** cases: After the axis is stopped, if necessary, the appropriate reversal point is approached and the distance-to-go deleted. The sparking-out strokes are then executed and the end position approached. Provided this has been set in OSCILL_CTRL_MASK.

The oscillation movement is then completed.

Note

During grinding, the calipers can be put into action via axial delete distance-to-go.

Emergency stop

The emergency stop completes the oscillation movement that must be restarted.

Reset

The oscillation movement is interrupted and deselected with a braking ramp. The options selected subsequently are not processed (sparking-out strokes, end point approach).

Working area limitation, limit switches

If it is detected during preprocessing that the oscillation movement would violate an active limitation, then an alarm is output and the oscillation movement not started.

If during an active oscillation movement the oscillation axis overtravels a limitation which has been activated in the meantime (e.g. 2nd software limit switch), then the axis is decelerated down along a ramp and an alarm indicated.

⚠ CAUTION

Protection areas

No protection areas act for a oscillation movement.

Follow-up mode

There is no difference to positioning axes.

End of program

If the axis is not controlled by the PLC, then the program end is not reached until the oscillation movement is terminated (reaction as for POSA:

Positioning across block boundaries).

If the axis is controlled by the PLC, then it continues to oscillate after program end.

Mode change

The following table shows the operating modes in which oscillation can be implemented. Changeover to an operating mode which allows oscillation does not affect the oscillation movement. Changeover to inadmissible operating modes is rejected with an alarm. It is not possible to traverse an axis in oscillation mode while applying control commands from the NC program or via operator inputs (JOG) simultaneously; an alarm is output if this is attempted. The following applies: The type of movement first started has priority.

Table 11- 4 Operating modes which allow oscillation

Operating mode	Allows oscillation
AUTO	Yes
MDA	Yes
MDA Repos	Yes
MDA Teachin	Yes
MDA Ref	No
JOG	Yes
JOG Ref	No
JOG Repos	Yes

Single-block processing

If the axis is not controlled by the PLC, then it responds to a single block in the same way as a positioning axis (POSA), i.e. it continues the movement.

Override

The override is specified by the:

NC/PLC interface

Axial override acts on the oscillation axis.

Programming

The override acts on oscillation axes in the same way as on positioning axes.

Block search

In Block Search the last valid oscillation function is registered and the machine data OSCILL_MODE_MASK is activated (default) accordingly, either directly after NC start (when approaching the start position after block search) or after reaching the start position after block search.

OSCILL_MODE_MASK Bit 0:

0: Oscillation starts after reaching the start position.

1: Oscillation starts immediately after NC start.

REORG

Reversal point 1 is always approached first before oscillation continues.

ASUB

The oscillation movement continues while an ASUB (asynchronous subprogram) is in progress.

11.3 Oscillation controlled by synchronized actions

General procedure

An asynchronous oscillation movement is coupled via synchronized actions with an infeed motion and controlled accordingly.

References:

Function Manual, Synchronized Actions

The following description concentrates solely on the motion-synchronous actions associated with the oscillation function.

Functions

The following function complexes can be implemented by means of the language tools described in detail below:

1. Infeed at the reversal point (see Section "Infeed at reversal point 1 or 2 (Page 656)").
2. Infeed at the reversal point range (see Section "Infeed in reversal point range (Page 657)").
3. Infeed in both reversal points (see Section "Infeed at both reversal points (Page 659)").
4. Stop the oscillation movement at the reversal point until infeed is terminated (see Section "Stop oscillation movement at the reversal point (Page 660)").
5. Enable oscillation movement (see Section "Oscillation movement restarting (Page 661)").

6. Do not start partial infeed too early (see Section "Do not start partial infeed too early (Page 662)").

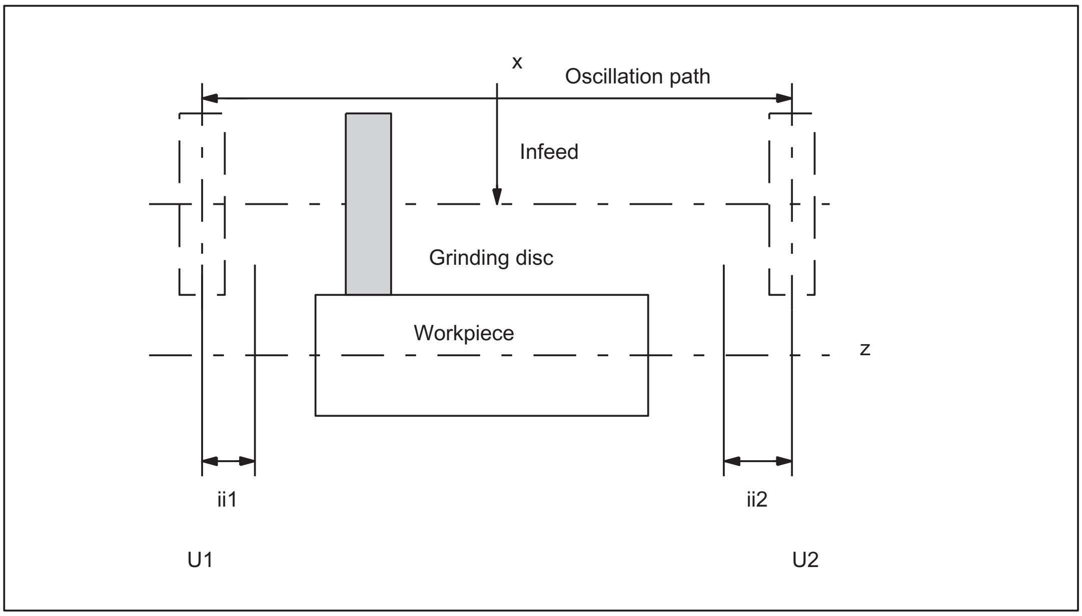


Figure 11-1 Arrangement of oscillation and infeed axes plus terms

Legend:

U1: Reversal point 1

U2: Reversal point 2

ii1: Reversal point range 1

ii2: Reversal point range 2

Programming

Before the motion block that contains the assignment between the infeed and the oscillating axis (see Section "Assignment of oscillation and infeed axes OSCILL (Page 662)"), the infeed definition (POSP) and the motion-synchronous actions, the parameters for the oscillation must first be defined (see Section "Influences on asynchronous oscillation (Page 643)"):

The oscillation axis is enabled via a WAITP [oscillation axis] (see MD30552 $MA_AUTO_GET_TYPE), allowing the oscillation parameters to be transferred, i.e. into the setting data, simultaneously. The symbolic names, e.g. SA43700 $SA_REVERSE_POS1 can then be used to program the motion-synchronous actions.

Note

For motion-synchronous actions with $SA_REVERSE_POS values, the comparison **values at the time of interpretation** are valid. If setting data is changed afterwards, this has no effect.

For motion-synchronous actions with $AA_REVERSE_POS values, the comparison values within the **interpoltation** are valid. This ensures a reaction to the modified reversal positions.

- **Motion-synchronous conditions WHEN, WHENEVER**
- Activation through motion block
  - Assign oscillation axis and infeed axes to one another OSCILL
  - Specify infeed response POSP.

The following sections present those elements which have not yet been dealt with.

Some examples are described in the "Examples" section.

Note

If the condition with which the motion-synchronous action (WHEN and WHENEVER) has been defined is no longer valid, the OVERRIDE for this condition is **automatically**set to 100% if the OVERRIDE had been set to 0% before.

Main run evaluation

It is possible to compare the synchronization conditions in the interpolation cycle in the main run with the current actual values ($$ variable on the right of comparison conditions). With normal system variable comparison, the expressions are evaluated in the first run. The complete extended possibilities for synchronized actions are listed in the following documentation:

References:

Function Manual, Synchronized Actions

Example 1

Oscillation, reversal position firmly set via setting data:

Program code	Comment
`$SA_OSCILL_REVERSE_POS1[Z]=-10`	
`$SA_OSCILL_REVERSE_POS2[Z]=10`	
`G0 X0 Z0`	
`WAITP(Z)`	
`ID=1 WHENEVER $AA_IM[Z] < $SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0`	
`ID=2 WHENEVER $AA_IM[Z] > $SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0`	`; If the actual value of the oscillation axis`
	`; has exceeded the reversal point,`
	`; the infeed axis is stopped.`
	`; Activate oscillation`
`OS[Z]=1 FA[X]=1000 POS[X]=40`	`; Deactivate oscillation`
`OS[Z]=0`	
`M30`	

Example 2

Oscillation with online change of the reversal position, i.e. any modification of reversal position 1 via the user surface are immediately taken into account with active oscillation movement.

Program code	Comment
`$SA_OSCILL_REVERSE_POS1[Z]=-10`	
`$SA_OSCILL_REVERSE_POS2[Z]=10`	
`G0 X0 Z0`	
`WAITP(Z)`	
`ID=1 WHENEVER $AA_IM[Z] < $$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0`	
`ID=2 WHENEVER $AA_IM[Z] > $$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0`	
	; If the actual value of the oscillation axis
	; has exceeded the reversal point,
	; the infeed axis is stopped.
`OS[Z]=1 FA[X]=1000 POS[X]=40`	; Activate oscillation
`OS[Z]=0`	; Deactivate oscillation
`M30`	

11.3.1 Infeed at reversal point 1 or 2

Function

As long as the oscillation axis has not reached the reversal point, the infeed axis does not move.

Application

Direct infeed in reversal point

Programming

For reversal point 1:

WHENEVER $AA_IM[Z] <> $SA_OSCILL_REVERSE_POS1[Z]

DO $AA_OVR[X] = 0 $AA_OVR[Z] = 100

For reversal point 2:

WHENEVER $AA_IM[Z] <> $SA_OSCILL_REVERSE_POS2[Z]

DO $AA_OVR[X] = 0 $AA_OVR[Z] = 100

Explanation of system variables:

$AA_IM[Z]: Current position of oscillating axis Z in the MCS

$SA_OSCILL_REVERSE_POS1[Z]: Position of the reversal point1 of the oscillation axis

$AA_OVR[X]: Axial override of the infeed axis

$AA_OVR[Z]: Axial override of the oscillation axis

Explanation of key words:

WHENEVER ... DO ... always when the condition is fulfilled, then ...

Infeed

The absolute infeed value is defined by the POSP instruction (see Section "Definition of infeeds POSP (Page 663)").

Assignment

The assignment between the oscillation axis and the infeed axis is defined by the OSCILL instruction (see Section "Assignment of oscillation and infeed axes OSCILL (Page 662)").

11.3.2 Infeed in reversal point range

Function

Reversal point range 1:

No infeed takes place provided the oscillation axis has not reached the reversal point range (position at reversal point 1 plus contents of variables ii1). This applies on the condition that reversal point 1 is set to a lower value than reversal point 2. If this is not the case, then the condition must be changed accordingly.

Application

Reversal point range 1:

The purpose of the synchronized action is to prevent the infeed movement from starting until the oscillation movement has reached reversal point range 1 (see "Figure 11-1 Arrangement of oscillation and infeed axes plus terms (Page 654)").

Programming

Reversal point range 1:

WHENEVER $AA_IM[Z] > $SA_OSCILL_REVERSE_POS1[Z] + ii1

DO $AA_OVR[X] = 0

Explanation of system variables:

$AA_IM[Z]: Current position of oscillating axis Z

$SA_OSCILL_REVERSE_POS1[Z]: Position of reversal point 1 of the oscillation axis

$AA_OVR[X]: Axial override of the infeed axis

ii1: Size of the reversal point range (number of variables)

Explanation of key words:

WHENEVER ... DO ... always when the condition is fulfilled, then ...

Function

Reversal point range 2:

The infeed axis stops until the current position (value) of the oscillation axis is lower than the position at reversal point2 minus the contents of variable ii2. This applies on condition that the setting for reversal point position 2 is higher than that for reversal point position 1. If this is not the case, then the condition must be changed accordingly.

Application

Reversal point range 2:

The purpose of the synchronized action is to prevent the infeed movement from starting until the oscillation movement has reached reversal point range 2 (see "Figure 11-1 Arrangement of oscillation and infeed axes plus terms (Page 654)").

Programming

Reversal point range 2:

WHENEVER $AA_IM[Z] < $SA_OSCILL_REVERSE_POS2[Z] - ii2

DO $AA_OVR[X] = 0

Explanation:

$AA_IM[Z]: Current position of oscillating axis Z

$SA_OSCILL_REVERSE_POS2[Z]: Position of reversal point 2 of the oscillation axis

$AA_OVR[X]: Axial override of the infeed axis

ii2: Magnitude of reversal range 2 (user variable)

Infeed

The absolute infeed value is defined by the POSP instruction (see Section "Definition of infeeds POSP (Page 663)").

Assignment

The assignment between the oscillation axis and the infeed axis is defined by the OSCILL instruction (see Section "Assignment of oscillation and infeed axes OSCILL (Page 662)").

See also

Oscillation controlled by synchronized actions (Page 653)

11.3.3 Infeed at both reversal points

General procedure

The functions described above for infeed at the reversal point and in the reversal point range can be freely combined.

Combinations

Infeed:

to U1 - to U2

to U1 - range U2

range U1 - to U2

range U1 - range U2

One-sided infeed

to U1

to U2

range U1

range U2

(See Section "Infeed at reversal point 1 or 2 (Page 656)" and "Infeed in reversal point range (Page 657)")

11.3.4 Stop oscillation movement at the reversal point

Function

Reversal point 1:

Every time the oscillation axis reaches reversal position 1, it must be stopped by means of the override and the infeed movement started.

Application

The synchronized action is used to hold the oscillation axis stationary until partial infeed has been executed. This synchronized action can be omitted if the oscillation axis need not wait at reversal point 1 until partial infeed has been executed. At the same time, this synchronized action can be used to start the infeed movement if this has been stopped by a previous synchronized action which is still active.

Programming

WHENEVER $AA_IM[oscillation axis] == $SA_OSCILL_REVERSE_POS1[oscillation axis]

DO $AA_OVR[oscillation axis] = 0 $AA_OVR[infeed axis] = 100

Explanation of system variables:

$AA_IM[oscillation axis]: Current position of oscillation axis

$SA_OSCILL_REVERSE_POS1[oscillation axis]: Reversal point 1 of the oscillation axis

$AA_OVR[oscillation axis]: Axial override of the oscillation axis

$AA_OVR[infeed axis]: Axial override of the infeed axis

Function

Reversal point 2:

Every time the oscillation axis reaches reversal position 2, it must be stopped by means of the override 0 and the infeed movement started.

Application

The synchronized action is used to hold the oscillation axis stationary until partial infeed has been executed. This synchronized action can be omitted if the oscillation axis need not wait at reversal point 2 until partial infeed has been executed. At the same time, this synchronized action can be used to start the infeed movement if this has been stopped by a previous synchronized action which is still active.

Programming

WHENEVER $AA_IM[oscillation axis] == $SA_OSCILL_REVERSE_POS2[oscillation axis]

DO $AA_OVR[oscillation axis] = 0 $AA_OVR[infeed axis] = 100

Explanation:

$AA_IM[oscillation axis]: Current position of oscillation axis

$SA_OSCILL_REVERSE_POS2[oscillation axis]: Reversal point 2 of the oscillation axis

$AA_OVR[oscillation axis]: Axial override of the oscillation axis

$AA_OVR[infeed axis]: Axial override of the infeed axis

11.3.5 Oscillation movement restarting

Function

The oscillation axis is started via the override whenever the distance-to-go for the currently traversed path section of the infeed axis = 0, i.e. partial infeed has been executed.

Application

The purpose of this synchronized action is to continue the movement of the oscillation axis on completion of the part infeed movement. If the oscillation axis need not wait for completion of partial infeed, then the motion-synchronous action with which the oscillation axis is stopped at the reversal point must be omitted.

Programming

WHENEVER $AA_DTEPW[infeed axis] == 0

DO $AA_OVR[oscillation axis] =100

Explanation of system variables:

$AA_DTEPW[infeed axis]: axial remaining travel distance for the infeed axis in the MCS: Path distance of the infeed axis

$AA_OVR[oscillation axis]: Axial override for oscillation axis

Explanation of key words:

WHENEVER ... DO ... always when the condition is fulfilled, then ...

11.3.6 Do not start partial infeed too early

Function

The functions described above prevent any infeed movement outside the reversal point or the reversal point range. On completion of an infeed movement, however, restart of the next partial infeed must be prevented.

Application

A channel-specific flag is used for this purpose. This flag is set at the end of the partial infeed (partial distance-to-go == 0) and is deleted when the axis leaves the reversal point range. The next infeed movement is then prevented by a synchronized action.

Programming

WHENEVER $AA_DTEPW[infeed axis] == 0

DO $AC_MARKER[Index]=1

and, for instance, for reversal point 1:

WHENEVER $AA_IM[Z] <> $SA_OSCILL_REVERSE_POS1[Z]

DO $AC_MARKER[Index]=0

WHENEVER $AC_MARKER[index]==1

DO $AA_OVR[infeed axis]=0

Explanation of system variables:

$AA_DTEPW[infeed axis]: axial remaining travel distance for the infeed axis in the MCS: Path distance of the infeed axis

$AC_MARKER[index]: Channel-specific marker with index

$AA_IM[oscillation axis]: Current position of oscillation axis

$SA_OSCILL_REVERSE_POS1[oscillation axis]: Reversal point 1 of the oscillation axis

$AA_OVR[infeed axis]: Axial override for infeed axis

Explanation of key words:

WHENEVER ... DO ... always when the condition is fulfilled, then ...

11.3.7 Assignment of oscillation and infeed axes OSCILL

Function

One or several infeed axes are assigned to the oscillation axis with command OSCILL. Oscillation motion starts.

The PLC is informed of which axes have been assigned via the NC/PLC interface. If the PLC is controlling the oscillation axis, it must now also monitor the infeed axes and use the signals for the infeed axes to generate the reactions on the oscillation axis via 2 stop bits of the interface.

Application

The axes whose response has already been defined by synchronous conditions are assigned to one another for activation of oscillation mode. The oscillation movement is started.

Programming

OSCILL[oscillation axis] = (infeed axis1, infeed axis2, infeed axis3)

Infeed axis2 and infeed axis3 in brackets plus their delimiters can be omitted if they are not required.

11.3.8 Definition of infeeds POSP

Function

The control receives the following data for the infeed axis:

- Total infeed
- Part infeed at reversal point/reversal point range
- Part infeed response at end

Application

This instruction must be given after activation of oscillation with `OSCILL` to inform the controller of the required infeed values at the reversal points/reversal point ranges.

Programming

POSP[infeed axis] = (end position, part section, mode)

End position: End position for the infeed axis after all partial infeeds have been traversed.

Part section: Part infeed at reversal point/reversal point range

Mode 0: For the last two part steps, the remaining path up to the target point is divided into two equally large residual steps (default setting).

Mode 1: The part length is adjusted such that the total of all calculated part lengths corresponds exactly to the path up to the target point.

11.3.9 External oscillation reversal

For example, keys on the PLC can be used to change the oscillation area or instantaneously reverse the direction of oscillation.

The edge-triggered PLC input signal DB31, ... DBX28.0 (oscillation reversal) is used to brake the current oscillation motion and then traverse in the opposite direction. The braking phase is signaled via the PLC output signal DB31, ... DBX100.2 (oscillation reversal active).

The braking position of the axis can be accepted as the **new reversal position** by means of the PLC signal DB31, ... DBX28.4 (change reversal position).

The PLC input signal DB31, ... DBX28.3 (select reversal point) is ignored, but rather the change affects the last initiated *external oscillation reversal* command.

No change in the reversal points applied via handwheel or JOG keys may be active for the relevant axis. If handwheel or JOG key changes are currently active, display alarm 20081 (Braking position cannot be accepted as reversal position - handwheel active) will be generated. The alarm is automatically reset when the conflict has been eliminated.

Hold time

No stop time is applied for a change of direction due to an *external oscillation reversal*. The axis waits for the exact stop fine signal. Any configured exact stop condition is fulfilled.

Infeed motion

With non-modal oscillation, no infeed movement is performed for a change of direction due to an *external oscillation reversal* as the reversal position has not been reached and consequently the appropriate synchronized action is not fulfilled.

System variables

The braking position can be scanned via system variable $AA_OSCILL_BREAK_POS1, when approach to reversal position 1 is aborted or via

$AA_OSCILL_BREAK_POS2 when approach to reversal position 2 is aborted.

If the relevant reversal point is approached again, the position of the reversal point can be scanned in $AA_OSCILL_BREAK_POS1 or $AA_OSCILL_BREAK_POS2.

In other words, only after an *External oscillation reversal* command is there a difference between the values in $AA_OSCILL_BREAK_POS1 and $AA_OSCILL_REVERSE_POS1 or the values in $AA_OSCILL_BREAK_POS2 and $AA_OSCILL_REVERSE_POS2.

External oscillation reversal can therefore be **detected** by a synchronized action (see Section "Examples (Page 665)").

Special cases

If the PLC input signal "oscillation reversal" is activated as the axis is approaching the start position, the approach movement is aborted and the axis continues by approach interruption position 1.

If the PLC input signal "oscillation reversal" is set during a stop period, the stop timer is deactivated; if exact stop fine has not yet been reached, the axis waits for the exact stop fine reached signal before continuing its motion.

If the PLC input signal "oscillation reversal" is activated as the axis is approaching the end position, the approach movement is aborted and oscillation is terminated.

For an example of the external oscillation reversal command (see Section "Change reversal position via synchronized action with "external oscillation reversal" (Page 676)").

11.4 Marginal conditions

Availability of the "Oscillation" function

The function is an option ("oscillation functions"), which must be assigned to the hardware through the license management.

11.5 Examples

Requirements

The examples given below require components of the NC language specified in the sections entitled:

- Asynchronous oscillation

and

- Oscillation controlled by synchronized actions.

11.5.1 Example of asynchronous oscillation

Task

The oscillation axis Z must oscillate between -10 and 10. Approach reversal point 1 with exact stop coarse and reversal point 2 without exact stop. The oscillation axis feedrate must be 5000. 3 sparking-out strokes must be executed at the end of the machining operation followed by approach by oscillation axis to end position 30. The feedrate for the infeed axis is 1000, end of the infeed in X direction is at 15.

Program section

Program code	Comment
OSP1[Z]=-10	; Reversal point 1
OSP2[Z]=10	; Reversal point 2
OST1[Z]=-1	; Stop time at reversal point 1: Exact stop coarse
OST2[Z]=-2	; Stop time at reversal point 2: without exact stop
FA[Z]=5000	; Infeed for oscillating axis
OSNSC[Z]=3	; three spark-out strokes
OSE[Z]=-3	; End position
OS1 F500 X15	; start oscillation, infeed X axis
	; with infeed 500, feed target 15

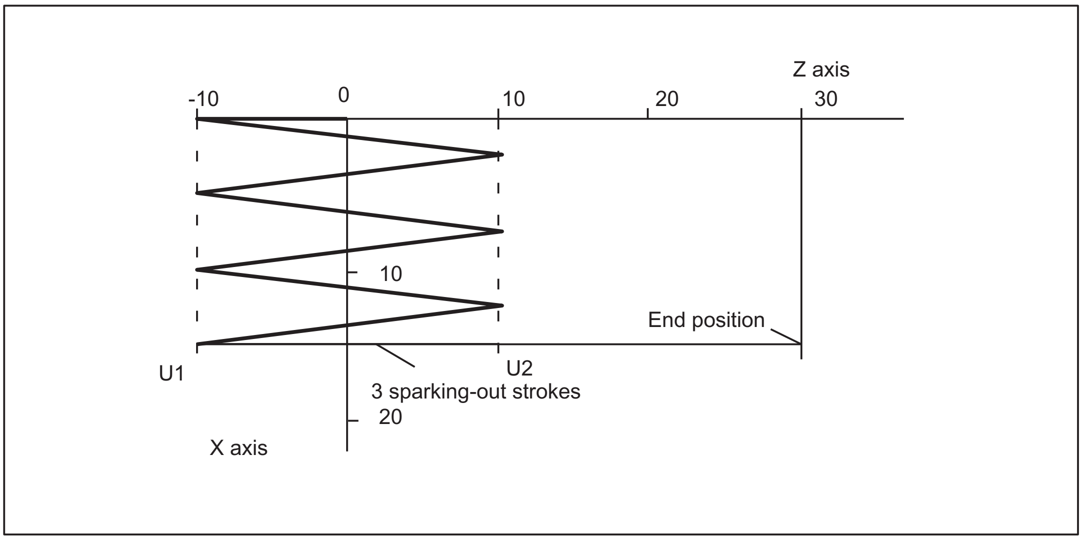


Figure 11-2 Sequences of oscillation movements and infeed, example 1

11.5.2 Example 1 of oscillation with synchronized actions

Task

Direct infeed must take place at reversal point 1; the oscillation axis must wait until the partial infeed has been executed before it can continue traversal. At reversal point 2, the infeed must take place at a distance of -6 from reversal point 2; the oscillation axis must not wait at this reversal point until partial infeed has been executed. Axis Z is the oscillation axis and axis X is the infeed axis (see Section "Oscillation controlled by synchronized actions (Page 653)).

Note

The setting data OSCILL_REVERSE_POS_1/2 are values in the machine coordinate system; therefore comparison is only suitable with $AA_IM[n].

Program section

```
Program code                 Comment
; Example 1: Oscillation with synchronized actions
OSP1[Z]=10 OSP2[Z]=60        ; explain reversal points 1 and 2
OST1[Z]=-2 OST2[Z]=0         ; Reversal point 1: without exact stop
                             ; Reversal point 2: Exact stop fine
FA[Z]=5000 FA[X]=250         ; Feed for oscillating axis, feedrate, infeed axis
OSCTRL[Z]=(1+8+16,0)         ; Switch off oscillation motion at reversal point 1
                             ; After delete DTG spark-out and approach end position
                             ; After DTG, approach relevant reversal position
                             ;
OSNSC[Z]=3                   ; 3 sparking-out strokes
OSE[Z]=0                     ; End position = 0
WAITP(Z)                     ; enable oscillation for the Z axis
;
; motion-synchronous actions:
;
; always, when                the current position of the oscillating axis in the
;                             Machine Coordinate System
; not equal to                reversal position 1
; then                        set the marker with index 1 to value 0
;                             (Reset marker 1)
WHENEVER $AA_IM[Z]<>$SA_OSCILL_REVERSE_POS1[Z] DO $AC_MARKER[1]=0
;
; always, when                the current position of the oscillating axis in the
;                             Machine Coordinate System
```

Program code	Comment
; less than	the beginning of reversal range 2
;	(here: reversal point 2 -6),
; then	set the axial override of the infeed axis to 0%.
; and	set the marker with index 2 to value 0
;	(Reset marker 2)
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0 $AC_MARKER[2]=0	
;	
; always, when	the current position of the oscillating axis in the
;	Machine Coordinate System
; equal to	reversal position 1,
; then	set the axial override of the oscillation axis to 0%
;	
; and	Set the axial override of the infeed axis to
;	100% (so that the previous synchronized action
;	is cancelled!)
WHENEVER $AA_IM[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[Z]=0 $AA_OVR[X]=100	
;	
; always, when	the distance-to-go of the partial infeed is
; equal to	0,
; then	set the marker with index 2 to value 1
; and	set the marker with index 1 to value 1
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[2]=1 $AC_MARKER[1]=1	
;	
; always, when	the flag with index 2 is
; equal to	1,
; then	Set the axial override of the infeed axis to
;	0%, this prevents a premature infeed
;	(oscillation axis has not exited the
;	reversal point 1).
;	
WHENEVER $AC_MARKER[2]==1 DO $AA_OVR[X]=0	
;	
; always, when	the flag with index 1 is
; equal to	1,
; then	Set the axial override of the infeed axis to
;	0%, this prevents a premature infeed
;	(oscillation axis has not exited the
;	reversal point range 2).
; and	Set the axial override of the infeed axis to
;	100% ('Start' oscillation)
WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100	
;	

Program code	Comment
; if the current position of the oscillating axis in the MCS is	
; equal to	reversal position 1,
; then	Set the axial override of the infeed axis to
;	100%
; and	Set the axial override of the infeed axis to
;	0% (so that the second synchronized action
;	is cancelled once!)
WHEN $AA_IM[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[Z]=100 $AA_OVR[X]=0	
;	
;--	
OSCILL[Z]=(X) POSP[X]=(5,1,1)	; Assign axis X to the oscillation axis Z as
	; infeed axis, this should
	; infeed to end position 5 in steps
	; from 1 and the sum of all part lengths should
	; give exactly the end position
M30	; End of program

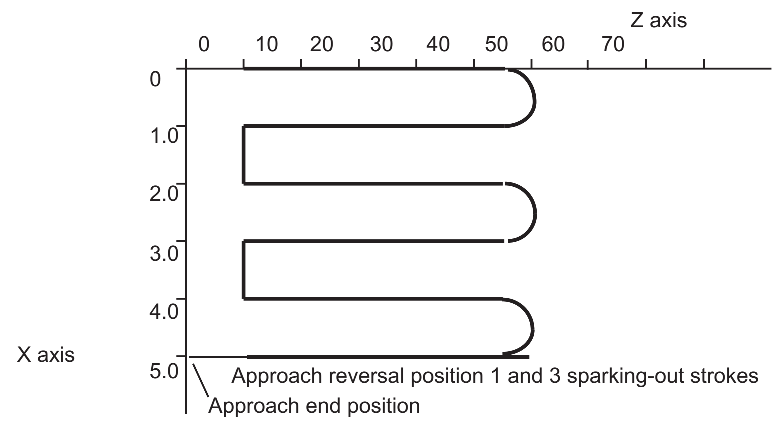


Figure 11-3 Sequences of oscillation movements and infeed, example 1

11.5.3 Example 2 of oscillation with synchronized actions

Task

No infeed must take place at reversal point 1. At reversal point 2, the infeed must take place at distance ii2 from reversal point 2; the oscillation axis must wait at this reversal point until partial infeed has been executed. Axis Z is the oscillation axis and axis X the infeed axis.

Program section

Example 2: Oscillation with synchronized actions

```
Program code             Comment
DEF INT ii2              ; Define variable for reversal area 2
;
OSP1[Z]=10 OSP2[Z]=60    ; explain reversal points 1 and 2
OST1[Z]=0 OST2[Z]=0      ; Reversal point 1: Exact stop fine
;                          Reversal point 2: Exact stop fine
FA[Z]=5000 FA[X]=100     ; Feed for oscillating axis, feed for infeed axis
OSCTRL[Z]=(2+8+16,1)     ; Switch off oscillation motion at reversal point 2
;                          After delete distance-to-go, spark-out and end position
;                          Approach appropriately after delete distance-to-go
;                          Approach reversal position
OSNSC[Z]=3               ; 3 sparking-out strokes
OSE[Z]=70                ; End position = 70
ii2=2                    ; Set reversal point range
WAITP(Z)                 ; permit oscillation for Z axis
;
; motion-synchronous actions:
; always, when             the current position of the oscillating axis in the
;                          Machine Coordinate System
; less than                the start of reversal area 2
; then                     set the axial override of the feed axis
;                          to 0%
; and                      set the marker with index 0 to value 0
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]-ii2 DO $AA_OVR[X]=0 $AC_MARKER[0]=0
;
; always, when             the current position of the oscillating axis in the
;                          Machine Coordinate System
; greater or equal to      reversal position 2
; then                     set the axial override of the oscillating axis
;                          to 0%
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0
;
```

```
Program code              Comment
; always, when              the distance-to-go of the partial infeed is
; equal to                  0,
; then                      set the marker with index 0 to value 1
WHENEVER $AA_DTEPW[X]  == 0 DO $AC_MARKER[0]=1
;
; always, when              the flag with index 0 is
; equal to                  1,
; then                      set the axial override of the infeed axis to 0%
;                           to prevent a new premature infeed
;                           (oscillation axis has not yet exited the reversal point
                            range 2,
;                           but the infeed axis is ready for a
;                           new infeed)
; and                       set the axial override of the oscillation axis to 100%
;                           (this cancels the 2nd synchronized action)
;
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
;
OSCILL[Z]=(X)             ; Start the axes
POSP[X]=(5,1,1)
;                           The oscillating axis Z is assigned axis X as
;                           an infeed axis
;                           Axis X should travel to end position 5 in
;                           increments of 1
;
M30
```

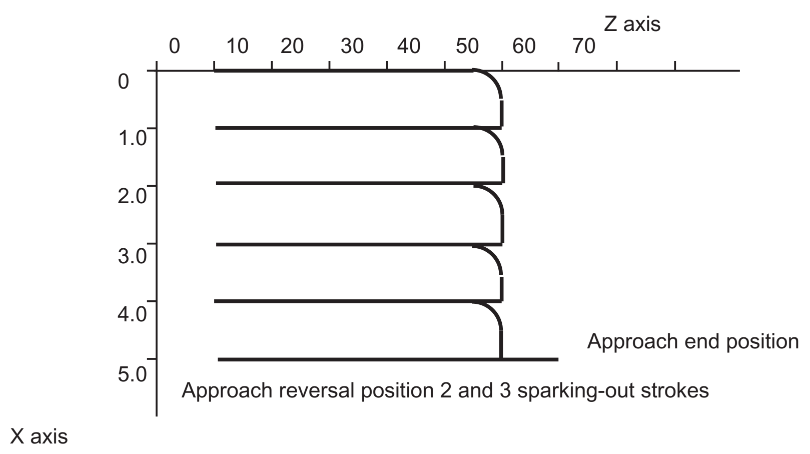


Figure 11-4 Sequences of oscillation movements and infeed, example 2

11.5.4 Examples for starting position

11.5.4.1 Define starting position via language command

Program code	Comment
WAITP(Z)	; enable oscillation for the Z axis
OSP1[Z]=10 OSP2[Z]=60	; explain reversal points 1 and 2
OST1[Z]=-2 OST2[Z]=0	; Reversal point 1: without exact stop
	; Reversal point 2: Exact stop fine
FA[Z]=5000 FA[X]=2000	; Infeed for oscillating axis,
	; feedrate for infeed axis
OSCTRL[Z]=(1+8+16,0)	; Switch off oscillation motion at reversal point 1
	; after delete DTG spark-out and approach end position,
	; After DTG, approach relevant reversal position
	;
OSNSC[Z]=3	; 3 sparking-out strokes
OSE[Z]=0	; End position = 0
OSB[Z]=0	; Start position = 0
OS[Z]=1 X15 F500	; Start oscillation, continuous infeed
OS[Z]=0	; Deactivate oscillation
WAITP(Z)	; wait for completion of the oscillation motion
M30	

Explanation

When the Z axis starts oscillation, it first approaches the starting position (position = 0 in the example) and then begins the oscillation motion between the reversal points 10 and 60. When the X axis has reached its end position 15, the oscillation finishes with 3 sparking out strokes and approach of end position 0.

11.5.4.2 Start oscillation via setting data

Program code	Comment
WAITP(Z)	
STOPRE	
$SA_OSCILL_REVERSE_POS1[Z]=-10	; reversal position 1 = -10
$SA_OSCILL_REVERSE_POS2[Z]=30	; reversal position 2 = 30
$SA_OSCILL_START_POS[Z] = -50	; Start position = -50
$SA_OSCILL_CTRL_MASK[Z] = 512	; Approach start position,
	; on switch-off, stop at the next
	; reversal point
	; do not approach any end position
	; no sparking-out strokes for DTG

Program code	Comment
`$SA_OSCILL_VELO[ Z ] = 5000`	; Infeed for oscillating axis
`$SA_OSCILL_IS_ACTIVE[ Z ] = 1`	; starting
`$SA_OSCILL_DWELL_TIME1[ Z ] = -2`	; without wait for exact stop
`$SA_OSCILL_DWELL_TIME2[ Z ] = 0`	; wait for fine exact stop
`STOPRE`	
`X30 F100`	
`$SA_OSCILL_IS_ACTIVE[ Z ] = 0`	; Stop
`WAITP(Z)`	
`M30`	

Description

When the Z axis starts oscillation, it first approaches the starting position (position = -50 in the example) and then begins the oscillation motion between the reversal points -10 and 30. When the X axis has reached its end position 30, the oscillation finishes at the next approached reversal point.

11.5.4.3 Non-modal oscillation (starting position = reversal point 1)

Oscillation with synchronized actions

Program code	Comment
`N701`	; Oscillation with synchronized actions,
	; start position == reversal point 1
;	
`N702 OSP1[Z]=10 OSP2[Z]=60`	; explain reversal points 1 and 2
`N703 OST1[Z]=0 OST2[Z]=0`	; Reversal point 1: Exact stop coarse
	; Reversal point 2: Exact stop fine
`N704 FA[Z]=5000 FA[X]=2000`	; Infeed for oscillating axis,
	; feedrate for infeed axis
`N705 OSCTRL[Z]=(1+8+16.0)`	; switch off oscillation motion at
	; reversal point 1 after DTG spark-out
	; and approach end position
	; After DTG, approach relevant
	; reversal position
	;
`N706 OSNSC[Z]=3`	; 3 sparking-out strokes
`N707 OSE[Z]=0`	; End position = 0
`N708 OSB[Z]=10`	; Start position = 10
`N709 WAITP(Z)`	; enable oscillation for the Z axis
;	

Program code	Comment
; motion-synchronous actions:	
; set marker with index 2 on 1 (initialization)	
WHEN TRUE DO $AC_MARKER[2]=1	
;	
; always, when	the marker with index 2 equals 0
;	and the current position of the oscillating axis
;	does not equal the reversal position 1
; then	set the marker with index 1 to 0.
;	
WHENEVER ($AC_MARKER[2] == 0) AND $AA_IW[Z]>$SA_OSCILL_REVERSE_POS1[Z]) DO $AC_MARKER[1]=0	
; always, when	the current position of the oscillating axis
;	is less than the start of reversal range 2
;	
; then	set the axial override of the feed axis
;	to 0 and set the marker with index 0 to
;	0
WHENEVER $AA_IW[Z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0 $AC_MARKER[0]=0	
;	
; always, when	the current position of the oscillating axis
;	is equal to the reversal position 1,
; then	set the axial override of the oscillating axis
;	to 0 and set the axial override of the
;	infeed axis to 100% (so that the
;	previous synchronized action is cancelled!)
WHENEVER $AA_IW[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[Z]=0 $AA_OVR[X]=100	
;	
; always, when	the distance-to-go of the partial infeed equals 0
;	
; then	Set the marker with index 0 to 1 and
;	set the marker with index 1 to 1 and
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1 $AC_MARKER[1]=1	
;	
; always, when	the marker with index 0 equals 1,
; then	set the axial override of the feed axis
;	to 0 to prevent a new premature
;	infeed!
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0	
;	

Program code	Comment
; always, when	the marker with index 1 equals 1,
; then	set the axial override of the feed axis
;	to 0 (to prevent a new premature
;	infeed!) and set the
;	axial override of the oscillation axis to 100%
;	(so that the previous
;	synchronized action is canceled!)
WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100	
;	
; When	the current position of the oscillating axis
;	is equal to the reversal position 1,
; then	reset the marker with index 2,
;	release the first synchronized action (no
;	infeed when reaching the start position
;	== reversal position 1)
WHEN $AA_IW[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AC_MARKER[2]=0	
;	
;----------------------------------	
N750 OSCILL[Z]=(X) POSP[X]=(5,1,1)	
; Assign axis X to the oscillation axis Z as infeed axis,	
; this should infeed to end position 5	
; in substeps of 1 and the sum of all sublengths	
; should be exactly the same as the end position.	
;	
N780 WAITP(Z)	; release the Z axis
;	
N790 X0 Z0	
N799 M30	; End of program

Description

The starting position matches reversal point 1. The WHEN .... synchronized actions (see above) prevent an infeed when the starting position is reached.

11.5.5 Example of external oscillation reversal

11.5.5.1 Change reversal position via synchronized action with "external oscillation reversal"

```
Program code                      Comment
DEFINE BREAKPZ AS $AA_OSCILL_BREAK_POS1[Z]
DEFINE REVPZ AS $SA_OSCILL_REVERSE_POS1[Z]

WAITP(Z)                          ; enable oscillation for the Z axis
OSP1[Z]=10 OSP2[Z]=60             ; explain reversal points 1 and 2
OSE[Z]=0                          ; End position = 0
OSB[Z]=0                          ; Start position = 0

                                  ; At external reversal of oscillation for
                                  ; oscillation reversal point 1, adapt this
WHENEVER BREAKPZ <> REVPZ DO $$SA_OSCILL_REVERSE_POS1 = BREAKPZ
OS[Z]=1 X150 F500                 ; Start oscillation, continuous infeed
OS[Z]=0                           ; Deactivate oscillation
WAITP(Z)                          ; wait for completion of the oscillation motion
M30
```

11.6 Data lists

11.6.1 Machine data

11.6.1.1 General machine data

Number	Identifier: $MN_	Description
10710	PROG_SD_RESET_SAVE_TAB	Oscillations to be saved from SD
11460	OSCILL_MODE_MASK	Control screen form for asynchronous oscillation

11.6.2 Setting data

11.6.2.1 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43700	OSCILL_REVERSE_POS1	Position at reversal point 1
43710	OSCILL_REVERSE_POS2	Position at reversal point 2
43720	OSCILL_DWELL_TIME1	Stop time at reversal point 1
43730	OSCILL_DWELL_TIME2	Stop time at reversal point 2
43740	OSCILL_VELO	Feed velocity of oscillation axis
43750	OSCILL_NUM_SPARK_CYCLES	Number of sparking-out strokes
43760	OSCILL_END_POS	Position after sparking-out strokes/at end of oscillation movement
43770	OSCILL_CTRL_MASK	Control screen form for oscillation
43780	OSCILL_IS_ACTIVE	Oscillation movement ON/OFF
43790	OSCILL_START_POS	Position that is approached after oscillation before reversal point 1, if activated in SD43770:

11.6.3 Signals

11.6.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
External oscillation reversal	DB31, ...DBX28.0	-
Set reversal point	DB31, ...DBX28.3	-
Alter reversal point	DB31, ...DBX28.4	-
Stop at next reversal point	DB31, ...DBX28.5	-
Stop along braking ramp	DB31, ...DBX28.6	-
PLC-controlled axis	DB31, ...DBX28.7	-

11.6.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Oscillation reversal is active	DB31, ...DBX100.2	-
Oscillation cannot start	DB31, ...DBX100.3	-
Error during oscillation movement	DB31, ...DBX100.4	-
Sparking-out active	DB31, ...DBX100.5	-

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Oscillation movement active	DB31, ...DBX100.6	-
Oscillation active	DB31, ...DBX100.7	-

11.6.4 System variables

11.6.4.1 Main run variables for motion-synchronous actions

Main run variable_read

The following variables are provided for main run variable_read:

$A_IN[<arith. expression>]	digital input (Boolean)
$A_OUT[<arith. expression>]	digital output (Boolean)
$A_INA[<arith. expression>]	analog input (Boolean)
$A_OUTA[<arith. expression>]	analog output (Boolean)
$A_INCO[<arith. expression>]	comparator inputs (Boolean)
$AA_IW[<axial expression>]	actual position PCS axis (real)
$AA_IB[<axial expression>]	actual position BCS axis (real)
$AA_IM[<axial expression>]	Actual position MCS axis (IPO setpoints) (real) With $AA_IM[S1] setpoints for spindles can be evaluated. Modulo calculation is used for spindles and rotary axes, depending on machine data $MA_ROT_IS_MODULO and $MA_DISPLAY_IS_MODULO.
$AA_OSCILL_BREAK_POS1	Breaking position after external oscillation reversal when approaching reversal point 1
$AA_OSCILL_BREAK_POS2	Breaking position after external oscillation reversal when approaching reversal point 2
$AC_TIME	Time from the start of the block (real) in seconds (including the times for the internally generated intermediate blocks)
$AC_TIMES	Time from the start of the block (real) in seconds (without times for the internally generated intermediate blocks)
$AC_TIMEC	Time from the start of the block (real) in IPO steps (including steps for the internally generated intermediate blocks)
$AC_TIMESC	Time from the start of the block (real) in IPO steps (without steps for the internally generated intermediate blocks)
$AC_DTBB	Distance from beginning of block in BCS (Distance to begin, baseCoor) (real)

$AC_DTBW	Distance from beginning of block in PCS (Distance to begin, workpieceCoor) (real)
$AA_DTBB[<axial expression>]	axial distance from beginning of block in BCS (Distance to begin, baseCoor) (real)
$AA_DTBW[<axial expression>]	axial distance from beginning of block in PCS (Distance to begin, workpieceCoor) (real)
$AC_DTEB	Distance to end of block in BCS (Distance to end) (Distance to end, baseCoor) (real)
$AC_DTEW	Distance to end of block in PCS (Distance to end, workpieceCoor) (real)
$AA_DTEB[<axial expression>]	axial distance to end of movement in BCS (Distance to end, baseCoor) (real)
$AA_DTEW[<axial expression>]	axial distance to end of movement in PCS (Distance to end, workpieceCoor) (real)
$AC_PLTBB	Distance from beginning of block in BCS (Path Length from begin, baseCoor) (real)
$AC_PLTEB	Distance to end of block in BCS (Distance to end) (Path Length to end, baseCoor) (real)
$AC_VACTB	Path speed in BCS (Velocity actual, baseCoor) (real)
$AC_VACTW	Path speed in PCS (Velocity actual, workPieceCoor) (real)
$AA_VACTB[<axial expression>]	Axis velocity in BCS (Velocity actual, baseCoor) (real)
$AA_VACTW[<axial expression>]	Axis velocity in PCS (Velocity actual, workPieceCoor) (real)
$AA_DTEPB[<axial expression>]	axial distance-to-go for oscillation infeed in BCS (Distance to end, pendulum, baseCoor) (real)
$AA_DTEPW[<axial expression>]	axial distance-to-go for oscillation infeed in PCS (Distance to end, pendulum, workpieceCoor) (real)
$AC_DTEPB	Path distance-to-go for oscillation infeed in BCS (not P2) (Distance to end, pendulum, baseCoor) (real)
$AC_DTEPW	Path distance-to-go for oscillation infeed in PCS (not P2) (Distance to end, pendulum, workpieceCoor) (real)
$AC_PATHN	(Path parameter normalized) (real) Normalized path parameter: 0 for beginning of block to 1 for end of block
$AA_LOAD[<axial expression>]	Drive utilization
$AA_POWER[<axial expression>]	Drive efficiency in W
$AA_TORQUE[<axial expression>]	Drive torque setpoint in Nm
$AA_CURR[<axial expression>]	Actual current value of axis

$AC_MARKER[<arithmetic_expression>] (int)	Flag variables: can be used to build complex conditions in synchronous actions: 8 Markers (Index 0 - 7) are available. Reset sets the markers to 0. Example: WHEN .....DO $AC_MARKER[0]=2 WHEN .....DO $AC_MARKER[0]=3 WHEN $AC_MARKER[0]==3 DO $AC_OVR=50 Can be read and written independently of synchronous actions in the parts program: IF $AC_MARKER == 4 GOTOF SPRUNG
$AC_PARAM[<arithmetic_expression>] (real)	Floating-decimal parameter for synchronous actions. Serves as intermediate saving and evaluation in synchronous actions. 50 Parameters (Index 0 - 49) are available.
$AA_OSCILL_REVERSE_POS1 [<axial expression>] (real)	
$AA_OSCILL_REVERSE_POS2 [<axial expression>] (real)	current oscillation reversal points 1 and 2: The current setting data from $SA_OSCILL_REVERSE_POS1 or $SA_OSCILL_REVERSE_POS2 is read. This enables setting data changes at the reversal positions during active oscillation, i.e. during an active synchronous action.

Conditions

Conditions for motion-synchronous actions are formulated:

Main run variable comparison operator expression

For details see:
References:
Function Manual Synchronized Actions

R2: Rotary axes 12

12.1 Brief Description

Rotary axes in machine tools

Rotary axes are used on many modern machine tools. They are required for tool and workpiece orientation, auxiliary movements and various other technological or kinematic purposes.

Typical examples for the use of rotary axes are the 5-axis milling machines. Only with the aid of rotary axes can the tip of the tool be positioned at any point on the workpiece for this type of machine.

Depending on the type of machine, many different demands are placed on a rotary axis. In order that the control can be adapted to the various types of machine, the individual rotary axis functions can be activated by means of machine data or special programming.

Rotary axes are always programmed in degrees. They are generally characterized by the fact that they assume the same position after exactly one rotation (modulo 360 degrees). Depending on the application in question, the traversing range of the rotary axis can be limited to less than 360 degrees (e.g., on swiveling axes for tool holders) or may be unlimited (e.g., when the tool or workpiece is rotated).

In many ways, the responses and features of rotary axes are identical to those of linear axes. The following description of functions is limited to a description of the special features of rotary axes and how they differ from linear axes.

Definition of a rotary axis

An axis can be declared as a rotary axis using the following axis-specific machine data:

MD30300 $MA_IS_ROT_AX

Geometry axes are defined as linear axes. Any attempt to define them as rotary axes will be rejected with alarm 4200 (Geometry axis cannot be defined as rotary axis).

Only when an axis has been declared as a rotary axis can it perform or use the functions described on the following pages (e.g., unlimited traversing range, modulo display of axis position, etc.).

Several axes can be declared as rotary axes simultaneously.

Types of rotary axis

Depending on the application, the operating range of a rotary axis can be unlimited (endlessly rotating in both directions [MD30310 $MA_ROT_IS_MODULE = 1]), limited by a software limit switch (e.g., operating range between 0° and 60°) or limited to an appropriate number of rotations (e.g., 1000°).

Some typical rotary-axis applications are listed below.

Typical applications

- 5-axis machining (operating range limited or unlimited)
- Rotary axis for eccentric machining (unlimited operating range)
- Rotary axis for cylindrical or form grinding (unlimited operating range)
- C axis with `TRANSMIT` (unlimited operating range)
- Rotary axis on winding machines (unlimited operating range)
- Rotary workpiece axis (C) on hobbing machines (unlimited operating range)
- Round tool magazines and tool turrets (unlimited operating range)
- Rotary axis for peripheral surface transformation (limited operating range)
- Swivel axes for gripping (operating range 360°)
- Rotary axes for swiveling (operating range < 360°; e.g., 60°)
- Milling swivel axis (A) on hobbing machines (operating range, e.g., 90°)

Axis addresses

Coordinate axes and directions of movement of numerically-controlled machine tools are designated according to DIN.

DIN 66025 specifies the following axis addresses for rotary or swivel axes:

- A, B and C with X, Y and Z as middle axis

  This means that A rotates about X, B rotates about Y and C rotates about Z (see fig.).
- The positive rotary-axis direction of rotation corresponds to a clockwise rotation when looking in the positive axis direction of the corresponding middle axis (see fig.).

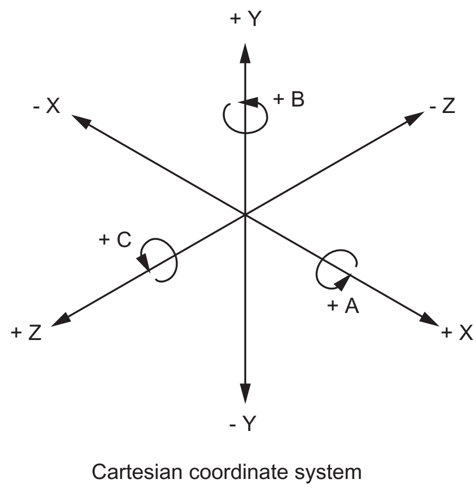


Figure 12-1 Axis identifiers and directions of movement for rotary axes

Extended addressing (e.g., C2=) or freely configured axis addresses can be used for additional rotary axes.

Note

Machine data MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB (assignment of geometry axis to channel axis) must be adapted to suit the corresponding axis.

Units of measurement

The following units of measurement apply as standard to inputs and outputs for rotary axes:

Units of measurement for rotary axes	
Physical quantity	Unit
Angular position	Degrees
Programmed angular velocity	Degrees/min
MD for angular velocity	[1] rev/min
MD for angular acceleration	[1] rev/sec^2
MD for angular jerk limitation	[1] rev/sec^3

[1] In the case of axis-specific machine data, these units are interpreted by the control as soon as the axis is declared as a rotary axis. The user can define other units for data inputs/outputs using machine data.

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

Operating range

The operating range can be defined by means of axis-specific machine and setting data (software limit switches and working-area limitations). As soon as modulo conversion is activated for the rotary axis (MD30310 $MA ROT_IS_MODULO = 1), the operating range is set to unlimited and the software limit switches and working-area limitations become inactive.

Using the following interface signal, software limit switches/working-area limitations can also be dynamically activated for modulo rotary axes by the PLC (where relevant, initiated from the part program using M/H functions):

DB31, ... DBX12.4 (modulo-limit enabled)

The feedback signal of the NC is realized using the interface signal:

DB31, ... DBX74.4 (modulo-limit enabled active)

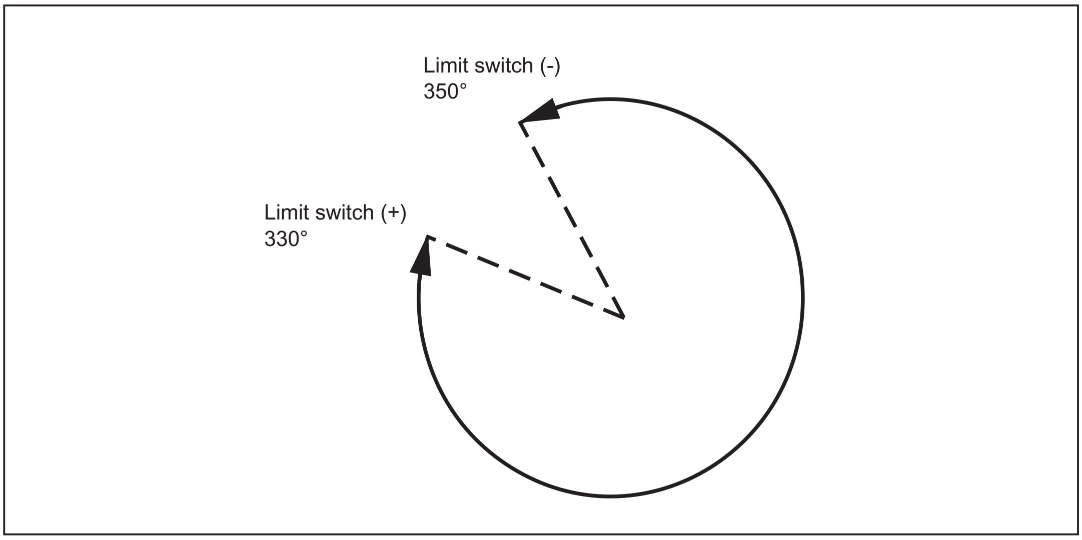


Figure 12-2 Limited operating area of a modulo rotary axis

Position display

The value range for the position display can be set to the modulo 360° representation, which is frequently selected for rotary axes:

MD30320 $MA_DISPLAY_IS_MODULO = 1

Feedrate

The programmed feedrate F corresponds to an angular velocity (degrees/min) in the case of rotary axes.

If rotary axes and linear axes traverse a common path with `G94` or `G95`, the feedrate should be interpreted in the linear-axis unit of measurement (e.g., mm/min, inch/min).

The tangential velocity of the rotary axis refers to diameter D_{unit} (unit diameter D_{unit}=360/π). In the case of unit diameter D=D_{unit}, the programmed angular velocity in degrees/min and the tangential velocity in mm/min (or inch/min) are numerically identical.

In general, the following applies for tangential velocity:

$F = F_{angle} * D/D_{unit}$	F	= Tangential velocity [mm/min]
	F_{angle}	= Angular velocity [degrees/min]
	D	= Diameter acted on by F [mm]
With $D_{unit} = 360/\pi$	D_{unit}=	Unit diameter [mm]
	π	= Circle constant Pi

Revolutional feedrate

In the JOG mode, the response of the axis/spindle also depends on the setting data:

SD41100 $SN_JOG_REV_IS_ACTIVE (revolutional feed rate for JOG active)

SD41100 $SN_JOG_REV_IS_ACTIVE	
Active	An axis/spindle is always traversed with revolutional feedrate: MD32050 $MA_JOG_REV_VELO (revolutional feedrate for JOG) or MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feedrate for JOG with rapid traverse override) , depending on the master spindle.
Not active	The behavior of the axis/spindle depends on the setting data: SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (revolutional feedrate for positioning axes/spindles)
	Behavior of a geometry axis on which a frame with rotation acts, depends on the channel-specific setting data: SD42600 $SC_JOG_FEED_PER_REV_SOURCE (in the JOG mode revolutional feed rate for geometry axes, on which the frame with rotation acts)

12.2 Modulo 360 degrees

Term "modulo 360°"

Rotary axes are frequently programmed in the 360° representation mode. The axis must be defined as a rotary axis in order to use the modulo feature.

With respect to a rotary axis, the term "modulo" refers to the mapping of the axis position within the control in the range 0° to 359.999°. With path defaults > 360° (e.g., for incremental programming using `G91`), the position is mapped in the range of values 0° to 360° following conversion within the control. Mapping is performed in JOG and AUTOMATIC modes. Exception: service display.

In the figure below, the rotary-axis absolute position in the positive direction of rotation is represented as a spiral. An arrow marks the actual absolute position on this spiral (example: point C' = 420°). By tracing the arrow back around the circle (position 0° of the spiral and circle are identical), it is possible to assign an appropriate modulo position within the 360° range to every absolute position. In the example below, absolute position point C' = 420° is mapped onto point C = 60° using modulo conversion.

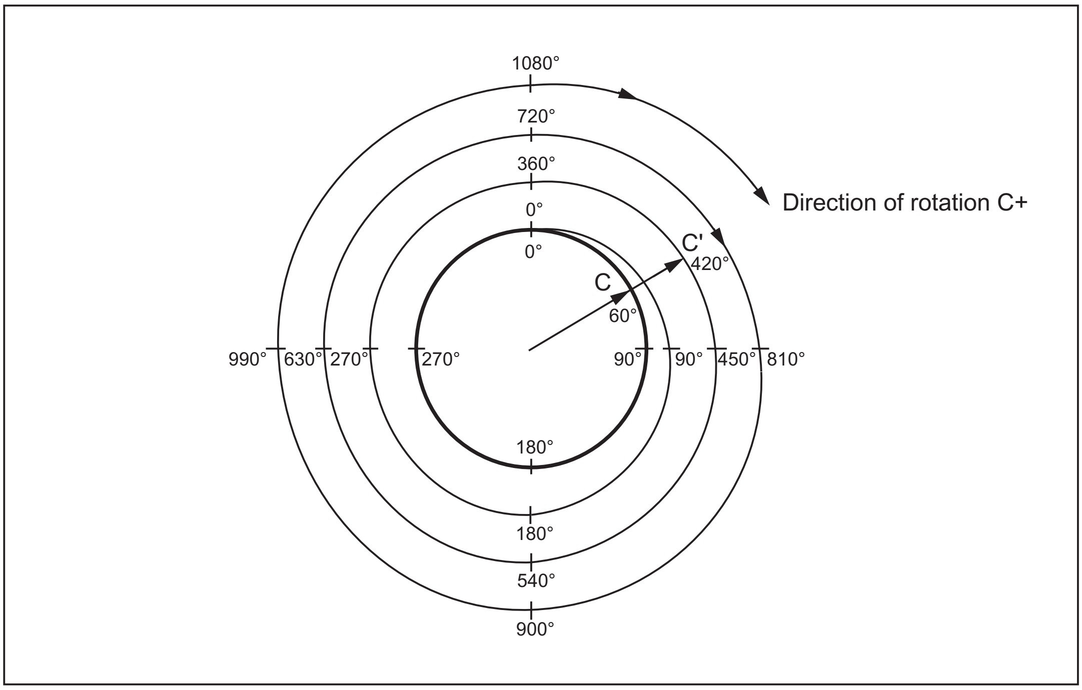


Figure 12-3 Modulo 360° map

Machine-data settings

Machine data can be used to define programming and positioning (MD30310 $MA_ROT_IS_MODULO) as well as the position display (MD30320 $MA_DISPLAY_IS_MODULO) individually in modulo 360° for each rotary axis, depending on the particular machine requirements.

Axis is modulo

MD30310 $MA_ROT_IS_MODULO = 1:

Activation of this machine data allows the special rotary-axis response to be utilized. The rotary-axis positioning response is thus defined during programming (`G90`, `AC`, `ACP`, `ACN` or `DC`). A modulo 360° representation is executed within the control after the current zero offsets have been taken into account. The resulting **target position within a revolution** is then approached.

The software limit switches and working-area limitations are inactive, meaning that the operating range is **unlimited** (continuously-turning rotary axis).

Modulo position display

MD30320 $MA_DISPLAY_IS_MODULO	
= 1	For rotary axes, a position display with "modulo 360°"(one revolution) is often required, i.e., with a positive direction of rotation the display is periodically reset within the control to 0.000° after 359.999° is reached, with a negative direction of rotation the positions are also displayed in the range 0° to 359.999°.
= 0	Absolute-position display would, in the case of a positive direction of rotation, for example, result in +360° being displayed after one revolution, +720° after two revolutions, etc, in contrast to the modulo 360° display. In this case, the display range is limited by the control in accordance with the linear axes.

MD30320 $MA_DISPLAY_IS_MODULO = 1

Note

The modulo 360° position display should always be selected for a modulo axis (MD30310 $MA_ROT_IS_MODULO = 1).

Starting position for the modulo rotary axis

A start position not equal to 0 for the modulo range can be defined:

MD30340 $MA_MODULO_RANGE_START (start position of the modulo range)

For example, this means that a modulo range of -180° to +180° can be achieved by entering -180 in MD30340.

The default setting of 0 (degrees) defines a modulo range of 0° - 360°.

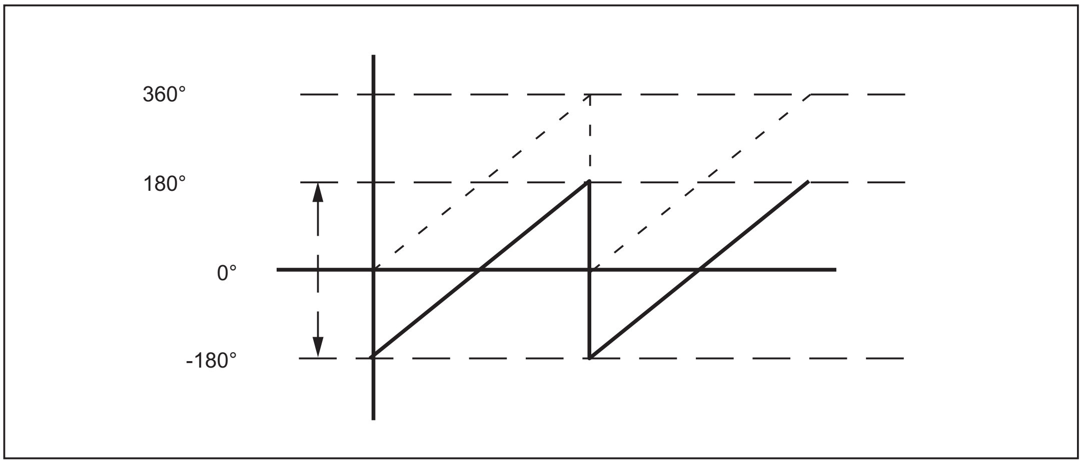


Figure 12-4 Starting position of -180° changes the modulo range to -180° to + 180°

Application

By approximating the two following machine data, indexing positions of modulo indexing axes can be implemented in the same way as for the modulo range (see also Section "T1: Indexing axes (Page 771)").

MD30503 $MA_INDEX_AX_OFFSET

MD30340 $MA_MODULO_RANGE_START

12.3 Programming rotary axes

12.3.1 General information

Note

General information about programming, see:
References:
Programming Manual Fundamentals

MD30310

Axis-specific machine data
MD30310 ROT_IS_MODULO (modulo conversion for rotary axis)
is used to define whether the rotary axis behaves as a linear axis during programming and positioning or whether rotary-axis special features are taken into account.

These features and any differences (mainly with respect to absolute programming) are explained on the following pages.

12.3.2 Rotary axis with active modulo conversion (continuously-turning rotary axis).

Activate modulo conversion

→ Set MD30310 $MA_ROT_IS_MODULO = 1.

Note

With modulo axes, it is advisable to set the position display to modulo 360º (set MD30320 $MA_DISPLAY_IS_MODULO = 1).

Absolute programming (AC, ACP, ACN, G90)

Example for positioning axis: POS[axis name] = ACP(value)

- The value identifies the rotary-axis target position in a range from 0° to 359.999°.

  Negative values are also possible if a range offset has been realized with the following machine data:

  MD30340 $MA_ MODULO_RANGE_START

  MD30330 MA_MODULO_RANGE
- `ACP` (positive) and `ACN` (negative) unambiguously define the rotary-axis traversing direction (irrespective of the actual position).
- When programming `AC` exclusively or with `G90`, the traversing direction depends on the rotary-axis actual position. If the target position is greater than the actual position, the axis traverses in the positive direction, otherwise it traverses in the negative direction.

  The positioning behavior can be configured via:

  MD30455 $MA_MISC_FUNCTION_MASK bit 2

  Bit 2 = 0: with `G90`, modulo axis positioned as standard using `AC`

  Bit 2 = 1: with `G90`, modulo axis positioned as standard using `DC` (shortest path)
- Use of ACP and ACN: With asymmetrical workpieces, it must be possible to define the traversing direction in order to prevent collisions during rotation.

Example:

C starting position is 0° (see figure below).

①	`POS[C] = ACP(100)`	Rotary axis C traverses to position 100° in the positive direction of rotation
②	`POS[C] = ACN(300)`	C traverses to position 300° in the negative direction of rotation
③	`POS[C] = ACP(240)`	C traverses to position 240° in the positive direction of rotation
④	`POS[C] = AC (0)`	C traverses to position 0° in the negative direction of rotation

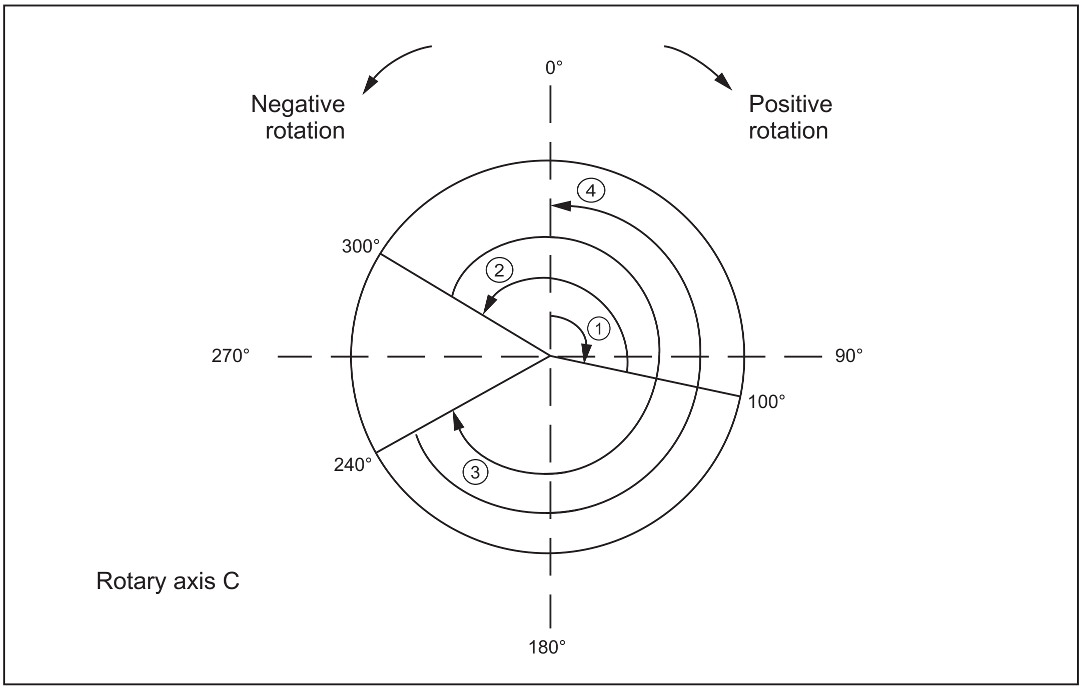


Figure 12-5 Examples of absolute programming for modulo axes

Absolute programming along the shortest path (DC)

POS[axis name] = DC(value)

- The value identifies the rotary-axis target position in a range from 0° to 359.999°. Alarm 16830, "Incorrect modulo position programmed", is output for values with a negative sign or ≥ 360°.
- With `DC` (Direct Control), the rotary axis approaches the programmed absolute position within one revolution along the **shortest path** (traversing movement max. ∓ 180°).
- The control calculates the direction of rotation and the traverse path according to the current actual position. If the path to be traversed is the same in both directions (180°), the positive direction of rotation receives preference.
- `DC` application example: the rotary table is required to approach the changeover position in the shortest time (and, therefore, via the shortest path) possible.
- If `DC` is programmed with a linear axis, alarm 16800, "DC traverse instruction cannot be used", is output.

Example:

C starting position is 0° (see figure below).

①	`POS[C] = DC(100)`	C axis traverses to position 100° along the shortest path
②	`POS[C] = DC(300)`	C axis traverses to position 300° along the shortest path
③	`POS[C] = DC(240)`	C axis traverses to position 240° along the shortest path
④	`POS[C] = DC(60)`	C axis traverses to position 60° along the shortest path. Since, in this case, the path is equal to 180° in both directions, preference is given to the positive direction of rotation.

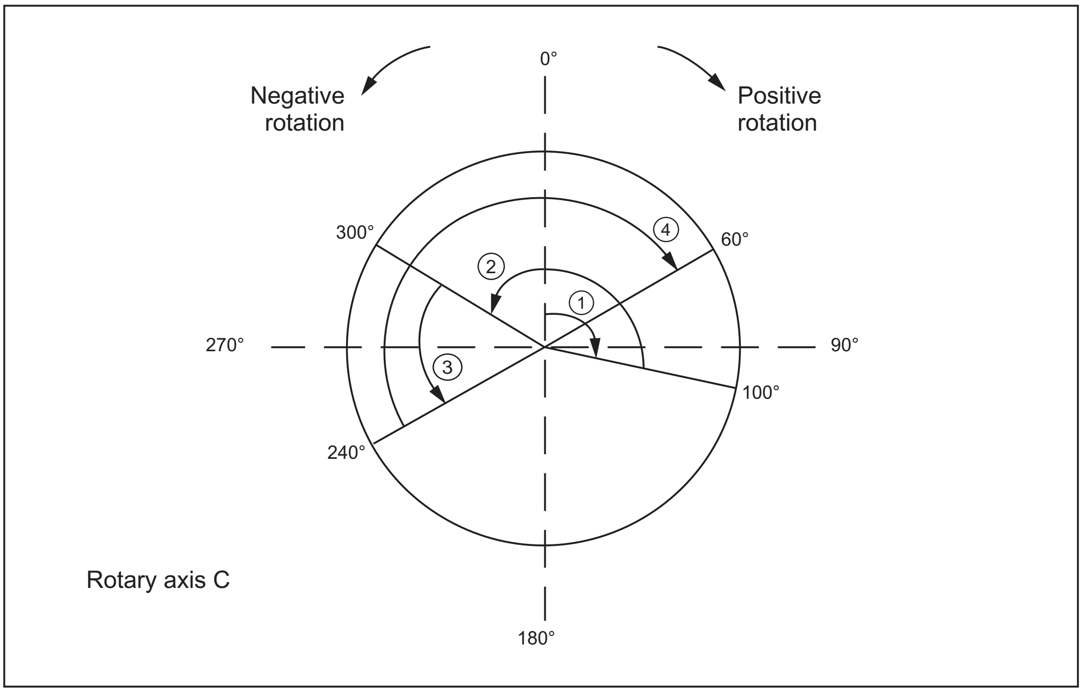


Figure 12-6 Examples of DC programming

Block-search response

After a block search with calculation, the collected search position of the modulo conversion can be interrogated using the $AC_RETPOINT system variable.

This system variable returns the position converted to modulo.

Supplementary conditions for ASUB after block search with calculation:

In this instance, as well as with the cross-channel block-search tool SERUPRO, the modulo conversion simulated in the block search must be performed in the part program.

Modulo rotary axis with/without working-area limitation

By setting the following interface signal for a modulo rotary axis, the working area limitation/software limit switch can be dynamically switched on/switched off by the PLC (similar to rotary axes):

DB31, ... DBX12.4

The actual state of the traversing range limitation is signaled back by the NC using the following interface signal:

DB31, ... DBX74.4

The monitoring function is activated if interface signal DB31, ... DBX12.4 was set by the PLC.

The M/H command, which causes the PLC to set the interface signal, must be followed by a STOPRE to ensure through synchronization that only the blocks after the changeover are monitored.

Supplementary conditions:

It is only possible to activate/deactivate software-limit-switch monitoring via the PLC interface for modulo axes.

Traversing-range monitoring for modulo axes can be implemented only if the axis is referenced and one limiting pair is active.

This always applies in the case of software limit switches, since these are always activated/deactivated in pairs. To monitor working area limitations correctly, **both** limitations must have been activated, either via `G26`/`G25` or the setting data:

SD43400 $SA_ WORKAREA_PLUS_ENABLE

and

SD43410 $SA_WORKAREA_MINUS_ENABLE.

Example of a traversing-range-limitation switchover

A pallet with several clamped workpieces should be machined successively on a modulo rotary axis. This pallet is then replaced by one with a built-on axis whose operating range must be monitored to prevent damage to supply lines.

Configuration:

MD30300 $MA_IS_ROT_AX[AX4] = 1

MD30310 $MA_ROT_IS_MODULO[AX4] = 1

MD36110 $MA_POS_LIMIT_PLUS[AX4] = 340

MD36100 $MA_POS_LIMIT_MINUS[AX4] = 350

Extract from part program:

Program code	Comment
M123	; Insert the pallet with quadruple clamping into the machine Deactivate the software limit switches on the B axis from the PLC DB35, DBX12.4=0
STOPRE	; Trigger a preprocessing stop
S1000 M3	
G4 F2	
G1 X0 Y300 Z500 B0 F5000	
CYCLE84(500,400,0,350,0,1,4,10,,0,500,1000) ; drilling cycle	
Z500	
B90	
CYCLE84(500,400,0,350,0,1,4,10,,0,500,1000) ; drilling cycle	
Z500	
B180	
CYCLE84(500,400,0,350,0,1,4,10,,0,500,1000) ; drilling cycle	
Z500	
B270	
CYCLE84(500,400,0,350,0,1,4,10,,0,500,1000) ; drilling cycle	
Z500	
G0 Z540 B0	
M124	; Insert the pallet with built-on axis into the machine Activate the software limit switches on the B axis from the PLC DB35, DBX12.4=1
	;
STOPRE	; Trigger a preprocessing stop
B270	

Incremental programming (IC, G91)

Example for positioning axis: **POS[axis name] = IC(+/-value)**

- The value identifies the rotary-axis traversing distance. The value can be negative and ≥ +/-360°.
- The value's **sign** unequivocally defines the rotary-axis **traversing direction**.
- Application example: milling a spiral groove across several revolutions

Example:

Programming	Effect
`POS[C]=IC(720)`	C axis traverses to 720° incrementally in the positive direction (two revolutions)
`POS[C]=IC(-180)`	C axis traverses to 180° incrementally in the negative direction

Endless traversing range

As soon as the modulo function is active, no limit is placed on the traversing range (software limit switches are not active). The rotary axis can now be programmed to traverse continuously.

Example:

```
Program code
LOOP:
POS[C] = IC(720)
GOTOB LOOP
```

12.3.3 Rotary axis without modulo conversion

Deactivate modulo conversion

→ Set MD30310 $MA_ROT_IS_MODULO = 0.

Absolute programming (AC, G90)

Example for positioning axis: **POS[axis name] = AC (+/-value)**

- The value and its sign uniquely identify the rotary-axis target position. The value can be ≥ +/-360°. The position value is limited by the software-limit-switch positions.
- The traversing direction is ascertained by the control according to the signed rotary-axis actual position.
- If `ACP` or `ACN` is programmed, alarm 16810, "ACP traverse instruction cannot be used", or alarm 16820 "ACN traverse instruction cannot be used", is output.
- Application example:

  Linear movements (cam gear) are linked to the rotary axis, thus certain end positions may not be overtraveled.

Example:

Programming	Effect
`POS[C] = AC (-100)`	Rotary axis C traverses to position -100°; traversing direction depends on the starting position
`POS[C] = AC (1500)`	Rotary axis C traverses to position 1500°

Absolute programming along the shortest path (DC)

POS[axis name] = DC(value)

Even if the rotary axis is not defined as a modulo axis, the axis can still be positioned with `DC` (Direct Control). The response is the same as on a modulo axis.

- The value identifies the rotary-axis target position **in a range from 0° to 359.999° (modulo 360°)**. Alarm 16830, "Incorrect modulo position programmed", is output for values with a negative sign or ≥ 360º.
- With DC (Direct Control), the rotary axis approaches the programmed absolute position within one revolution along the **shortest path** (traversing movement max. ± 180°).
- The control calculates the direction of rotation and the traverse path according to the current actual position (in relation to modulo 360°). If the path to be traversed is the same in both directions (180°), the positive direction of rotation receives preference.
- `DC` application example: the rotary table is required to approach the changeover position in the shortest time (and, therefore, via the shortest path) possible.
- If `DC` is programmed with a linear axis, alarm 16800, "DC traverse instruction cannot be used", is output.

Example:

Programming	Effect
`POS[C] = AC (7200)`	Rotary axis C traverses to position 7200°; traversing direction depends on the starting position
`POS[C] = DC(300)`	Rotary axis C approaches "modulo" position 300° along the shortest path Thus, C traverses about 60° with a negative direction of rotation and stops at absolute position 7140°.
`POS[C] = AC (7000)`	Rotary axis C traverses to position 7000° absolutely, so C traverses about 140° with a negative direction of rotation

---

Note

In this example, it would be advisable to activate the modulo 360º display (MD30320 $MA_DISPLAY_IS_MODULO = 1).

Incremental programming (IC, G91)

Example for positioning axis: POS[axis name] = IC(+/-value)

When programming with incremental dimensions, the rotary axis traverses across the same path as with the modulo axis. In this case, however, the traversing range is limited by the software limit switches.

- The value identifies the rotary-axis traversing distance.

  The value can be negative and ≥ +/-360°.
- The value's sign unequivocally defines the rotary-axis traversing direction.

Limited traversing range

The traversing range is limited as with linear axes. The range limits are defined by the "plus" and "minus" software limit switches.

12.3.4 Other programming features relating to rotary axes

Offsets

`TRANS` (absolute) and `ATRANS` (additive) can be applied to rotary axes.

Scalings

`SCALE` or `ASCALE` are not suitable for rotary axes, since the control always bases its modulo calculation on a 360º full circle.

Preset actual value memory

`PRESETON` is possible.

Indexing axes

See Section "T1: Indexing axes (Page 771)".

12.4 Activating rotary axes

Procedure

The procedure for activating rotary axes is the same as that for linear axes with a small number of exceptions. It should be noted that, as soon as the axis is defined as a rotary axis (MD30300 $MA_IS_ROT_AX = 1), the axis-specific-machine-/setting-data units are interpreted by the control as follows:

Positions	In "degrees"
Velocities	In "rev/min"
Accelerations	In "rev/sec^2"
Jerk limitation	In "rev/sec^3"

Special machine data

Special rotary-axis machine data may also have to be entered, depending on the application:

MD30310 $MA_ROT_IS_MODULO	Modulo conversion for positioning and programming
MD30320 $MA_DISPLAY_IS_MODULO	Modulo conversion for position display
MD10210 $MN_INT_INCR_PER_DEG	Computational resolution for angular positions

The following overview lists the possible combinations of these machine data for a rotary axis:

Possible combinations of rotary-axis machine data				
MD30300	MD30310	MD30320	Application permitted	Comment
0	0	0	Yes	The axis is a linear axis (default).
1	0	0	Yes	The axis is a rotary axis; modulo conversion is not used for positioning, i.e. the software limit switches are active; the position display is absolute.
1	0	1	Yes	The axis is a rotary axis; modulo conversion is not used for positioning, i.e. the software limit switches are active; the position display is modulo; Application: for axes with an operating range of +/-1000°, for example

Possible combinations of rotary-axis machine data				
1	1	1	Yes	The axis is a rotary axis; positioning is performed with modulo conversion, i.e. the software limit switches are inactive, the operating range is unlimited; the position display is modulo (setting most frequently used for rotary axes); axis with/without working-area limitation can be used.
1	1	0	Yes	The axis is a rotary axis; positioning is performed with modulo conversion, i.e. the software limit switches are inactive, the operating range is unlimited; the position display is absolute; axis with/without working-area limitation can be used.
0	0 or 1	0 or 1	Not recommended	Axis is not a rotary axis; therefore, the other MD are not evaluated.

JOG velocity for rotary axes

SD41130 $SN_JOG_ROT_AX_SET_VELO (JOG speed for rotary axes)

The above setting data can be used to define a valid JOG velocity for all rotary axes (see also Section "H1: Manual and handwheel travel (Page 143)").

If a value of 0 is entered in the setting data, the following axial machine data acts as JOG velocity for the rotary axis:

MD21150 $MC_JOG_VELO (conventional axis velocity)

12.5 Special features of rotary axes

Software limit switch

The software limit switches and working-area limitations are active and are required for swivel axes with a limited operating range. However, in the case of continuously rotating rotary axes (MD30310 $MA_ROT_IS_MODULO = 1), the software limit switches and working area limitations can be deactivated for individual axes.

A modulo rotary axis with/without working-area limitation can be used.

References:
Function Manual, Basic Functions; Axis Monitoring, Protection Zones (A3)

Mirroring of rotary axes

Mirroring can be implemented for rotary axes by programming `MIRROR(C)` or `AMIRROR(C)`.

Reference point approach

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

Spindles as rotary axes

For notes concerning the use of spindles as rotary axes (C axis operation), please refer to:

References:
Function Manual Basic Functions; Spindles (S1)

12.6 Examples

Fork head, inclined-axis head

Rotary axes are frequently used on 5-axis milling machines to swivel the tool axis or rotate the workpiece. These machines can position the tip of a tool on any point on the workpiece and take up any position on the tool axis. Various milling heads are required, depending on the application. The figure shows a fork head and an inclined-axis head as example rotary-axis arrangements.

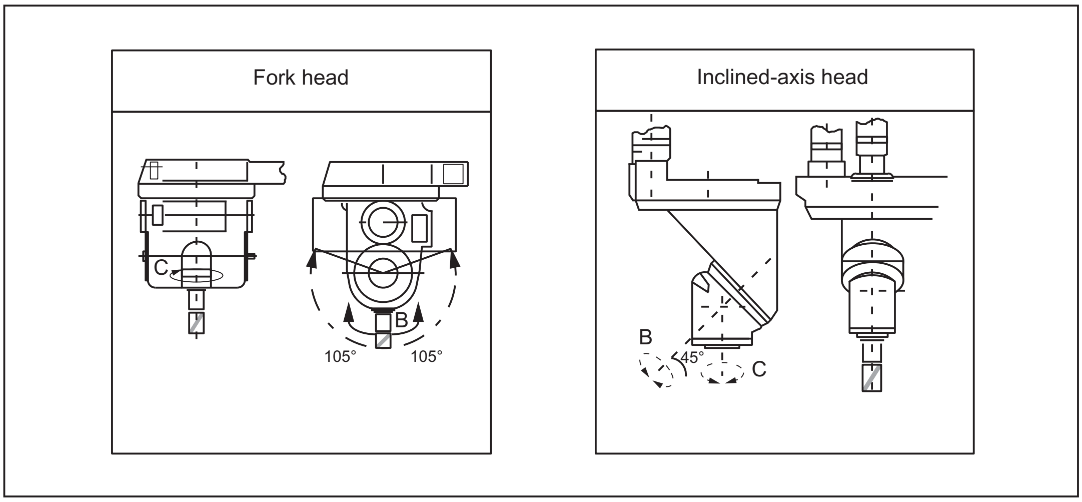


Figure 12-7 Fork head, inclined-axis head

12.7 Data lists

12.7.1 Machine data

12.7.1.1 General machine data

Number	Identifier: $MN_	Description
10210	INT_INCR_PER_DEG	Computational resolution for angular positions

12.7.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Axis is rotary axis
30310	ROT_IS_MODULO	Modulo conversion for rotary axis
30320	DISPLAY_IS_MODULO	Modulo actual-value display
30330	MODULO_RANGE	Modulo-range magnitude
30340	MODULO_RANGE_START	Modulo-range starting position
30455	MISC_FUNCTION_MASK	Axis functions
36100	POS_LIMIT_MINUS	Minus software limit switch
36110	POS_LIMIT_PLUS	Plus software limit switch

12.7.2 Setting data

12.7.2.1 General setting data

Number	Identifier: $SN_	Description
41130	JOG_ROT_AX_SET_VELO	JOG velocity for rotary axes

12.7.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43420	WORKAREA_LIMIT_PLUS	Plus working-area limitation
43430	WORKAREA_LIMIT_MINUS	Minus working-area limitation

12.7.3 Signals

12.7.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Traversing-range limitation for modulo axis	DB31, ...DBX12.4	DB380x.DBX1000.4

12.7.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Status of software-limit-switch monitoring for modulo axis	DB31, ...DBX74.4	DB390x.DBX1000.4

S3: Synchronous spindle 13

13.1 Brief description

13.1.1 Function

The "Synchronous spindle" function can be used to couple two spindles with synchronous position or speed. One spindle is defined as leading spindle (LS), the second spindle is then the following spindle (FS).

Speed synchronism:	$n_{FS} = k_{\ddot{U}} * n_{LS}$	with $k_{\ddot{U}}$ = 1, 2, 3, ...
Position synchronism:	$\varphi_{FS} = \varphi_{LS} + \Delta\varphi$	with 0° x Δφ t = 360°

Possible applications

Rear side machining

One application option is, for example, the reverse side machining in a double-spindle lathe with on-the-fly transfer of the workpiece from the position-synchronous LS to the FS (or vice versa), without having to decelerate down to standstill.

Multi-edge machining (polygonal turning)

The "Synchronous spindle" function provides the basis for multi-edge machining (polygonal turning) through specification of an integer gear ratio $k_{\ddot{U}}$ between LS and FS.

Number of FS

The number of FS's that can be operated synchronously to an LS is only restricted by the performance capability of the NC used. In principle, any number of FS can be coupled simultaneously to an LS in arbitrary channels of the NC.

2 pairs of synchronous spindles can be active simultaneously in each NC channel.

Definition

The assignment of FS to LS pair of synchronous spindles can be parameterized channel-specifically via machine data or flexibly defined via part program commands.

Selecting/de-selecting

Part program commands are used to select/deselect the synchronous operation of a pair of synchronous spindles.

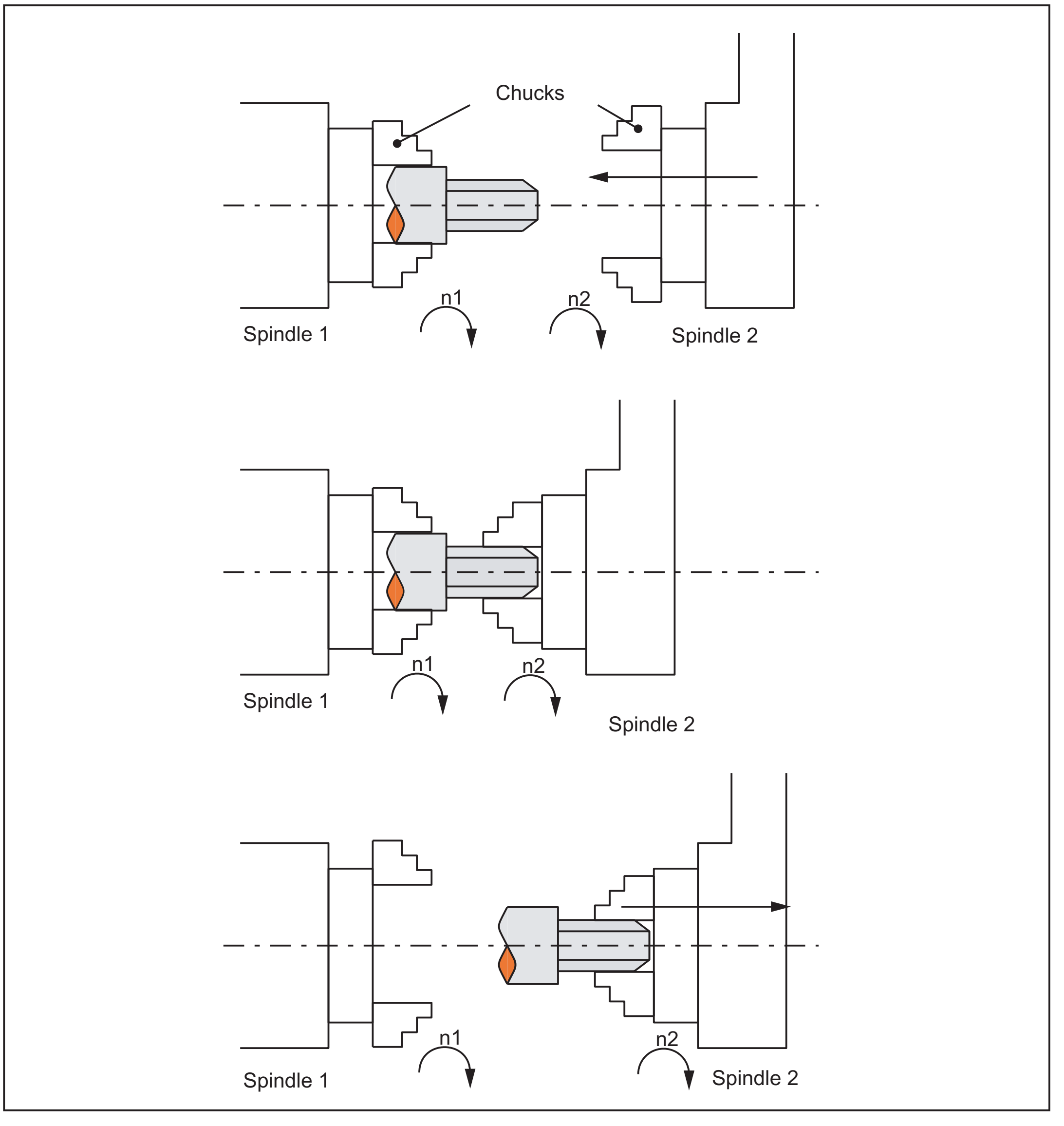


Figure 13-1 Synchronous operation: On-the-fly workpiece transfer from spindle 1 to spindle 2

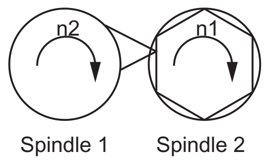


Figure 13-2 Synchronous operation: Polygonal turning

13.1.2 Synchronous mode

Description

<axial expression>:	can be: - Axis name - Spindle name
<axis name>:	C (if spindle has the name "C" in axis operation.)
<spindle name>:	Sn, SPI(n) where n = spindle number
<Spindle number>:	1, 2, ... according to the spindle number defined in MD35000 $MA_SPIND_ASSIGN_TO_MACHAX
(FS, LS, offset):	LS = Leading Spindle, FS = Following Spindle,Offset = read programmable offset of following spindle using system variables
$P_COUP_OFFS[Sn]	Programmed position offset of the synchronous spindle

Synchronous spindle pair

Synchronous operation involves a following spindle (FS) and a leading spindle (LS), referred to as the **synchronous spindle pair**. The following spindle imitates the movements of the leading spindle when a coupling is active (synchronous operation) in accordance with the defined functional interrelationship.

Synchronous mode

Synchronous mode (also referred to as “Synchronous spindle operation”) is another spindle operating mode. Before synchronous mode is activated, the following (slave) spindle must have been switched to position control. Synchronous operation is activated for the following spindle when the coupling is activated. As soon as the coupling is deactivated, the following spindle switches back to open-loop control mode.

As soon as synchronous operation is active for the following spindle, the following interface signal is reported to the PLC:

IS "Synchronous mode" (DB31, ... DBX84.4) = 1.

Number of synchronous spindles

It is possible to couple several following spindles to one leading spindle. The number of following spindles on this leading spindle depends on the respective version of the appropriate software versions.

Any number of following spindles in any channels of one NCU or a different NCU can be coupled to this leading spindle.

Note that one spindle is always the master and the number of couplings results from the number of axes less the master.

Options in synchronous mode

The following functions are available for synchronous mode:

- FS and LS turn at the same speed

  ($n_{FS} = n_{LS}$; transformation ratio $k_Ü = 1$)
- Rotation in the same or opposite direction between LS and FS

  (can be defined positively or negatively using transformation ratio $k_Ü$)
- Following and leading spindles rotate at different speeds

  ($n_{FS} = k_Ü \cdot n_{LS}$; transformation ratio $k_Ü \neq 1$)

  Application: Polygonal turning
- Settable angular position between FS and LS ($\varphi_{FS} = \varphi_{LS} + \Delta\varphi$)

  The spindles run at synchronous speed with a defined angular offset between FS and LS (position synchronous coupling).

  Application: Shaped workpieces
- Activation of synchronous operation between LS and FS can take place when the spindles are in motion or at standstill.
- The full functionality of the open-loop and position control modes is available for the leading spindle.
- When synchronous mode is not active, the FS and LS can be operated in all other spindle modes.
- The transformation ratio can also be altered when the spindles are in motion in active synchronous mode.
- With synchronous spindle coupling switched on, the offset of the FS to the LS (overlaid movement) can be altered.

Coupling options

Synchronous spindle couplings can be defined as

- Permanently configured via channel-specific machine data
  (hereinafter referred to as **"permanently configured coupling"**) as well as
- Freely defined using language commands (COUP...) in the part program
  (hereafter referred to as **"user defined coupling"**)

. The following variants are possible:

1. A fixed configuration for a coupling can be programmed via machine data. In addition, a second coupling can be freely defined via the part program.
2. No coupling is configured via machine data. In this case, the couplings can be user-defined and parameterized via the part program.

Separate following spindle interpolator

The separate **following spindle interpolator** allows a number of following spindles from different channels or from another NCU to be coupled as defined by the user to a single leading spindle. The following spindle interpolator is

- `COUPON` or `COUPONC` activated and
- `COUPOF` or `COUPOFS` deactivated

and is always located in the channel in which the `COUPON`, `COUPONC` command has been programmed for the following spindle. If the following spindle to be activated was previously programmed in another channel, `COUPON`/`COUPONC` initiates an axis replacement and fetches the spindle into its own channel.

Certain synchronous spindle functions can be controlled from the PLC by means of coupling-specific axial VDI interface signals. These signals act exclusively on the slave spindles and do not affect the leading spindle (see Section "Controlling synchronous spindle coupling via PLC (Page 716)").

Definition of synchronous spindles

Before synchronous operation is activated, the spindles to be coupled (FS, LS) must be defined.

This can be done in two ways depending on the application in question:

1. Permanently configured coupling:

   Machine axes that are to function as the following spindle (FS) and leading spindle (LS) are defined in channel-specific MD 21300 $MC_COUPLE_AXIS_1[n].

   The machine axes programmed as the LS and FS for this coupling configuration cannot be altered by the NC part program.

   If necessary, the coupling parameters can be modified with the NC part program.

2. User-defined coupling:

   Couplings can be created and altered in the NC part program with language command "COUPDEF(FS, LS, ...)". If a new coupling relationship is to be defined, it may be necessary to delete an existing user-defined coupling beforehand (with language command COUPDEL(FS, LS)).

   The axis names (Sn, SPI(n)) for the following and leading spindles must be programmed with FS and LS for every language command COUP..., thus ensuring that the synchronous spindle coupling is unambiguously defined.

   The valid spindle number must then be assigned axis-specific machine data of a machine axis:

   MD35000 $MA_SPIND_ASSIGN_TO_MACHAX.

IS "Following spindle active" (DB31, ... DBX99.1) and IS "Leading spindle active" (DB31, ... DBX99.0) indicate to the PLC for each machine axis whether the axis is active as a leading or following spindle.

The LS can be programmed either via a part program, PLC or also using synchronized actions.

Transformation ratio

The transformation ratio is programmed with separate numerical values for numerator and denominator (transformation ratio parameters). It is therefore possible to specify the transformation ratio very exactly, even with rational numbers.

In general:

$k_{Ü}$ = transformation ratio parameter for numerator Transformation ratio parameters for denominator = $Ü_{numerator}$: $Ü_{denominator}$

The value range of the transformation ratio parameter ($Ü_{numerator}$, $Ü_{denominator}$) is virtually unlimited in the control.

The transformation ratio parameters for the coupling configured via machine data can be defined in channel-specific SD42300: COUPLE_RATIO_1[n]. In addition, the transformation ratio can be altered with language command COUPDEF(FS, LS, $Ü_{numerator}$, $Ü_{denominator}$,...). The values entered in the setting data is not overwritten in this case (default settings).

The transformation ratio for the coupling defined via the NC part program can only be input with language command COUPDEF (...).

The new transformation ratio parameters take effect as soon as the COUPDEF command has been processed.

For further programming commands for synchronous spindle couplings, please see "Programming of synchronous spindle couplings" Section .

Coupling characteristics

The following characteristics can be defined for every synchronous spindle coupling:

- Block change behavior

  The condition to be fulfilled for a block change can be defined on activation of synchronous operation or on alteration of the transformation ratio or the defined angular offset when the coupling is active:

  – Block change takes place immediately

  – Block change in response to "Fine synchronism"

  – Block change in response to "Coarse synchronism"

  – Block change for IPOSTOP (i.e. after setpoint-end synchronism)

  – Check of the synchronism conditions at an arbitrary moment with `WAITC`.

- Type of coupling between FS and LS

  The position setpoint or the actual position value of the leading spindle can be used as the reference value for the following spindle. The following coupling types can therefore be selected:

  – Setpoint coupling (DV)

    Use in position controlled operation. The control dynamic response of both spindles should coincide as far as possible. Preferably, the setpoint coupling should be used.

  – Actual value coupling (AV)

    Application if no position control of the LS is possible or with great deviation of the control characteristics between FS and LS. The setpoints for the FS are derived from the actual values of the LS. The quality of synchronism is worse with a varying spindle speed than with the setpoint coupling.

  – Speed coupling (VV)

    Internally, the velocity coupling is a setpoint coupling. The requirements for FS and LS are lower. Position control and measuring systems are not required for FS and LS.

    The position offset between FS and LS is undefined.

The selection of the relevant coupling characteristics for the **configured coupling** is made using machine data (see Section "Configuration (Page 728)") and for the **user defined coupling** using the `COUPDEF` language command (see Section "Definition (COUPDEF) (Page 721)").

In addition, coupling characteristics Type of coupling and Block change response can be altered for the permanently configured coupling by means of language command `COUPDEF`.

References:

Programming Manual, Production Planning ("Synchronous Spindles").

Change protection for coupling characteristics

The channel-specific MD21340 $MC_COUPLE_IS_WRITE_PROT_1 is used to define whether or not the configured coupling parameters Transformation ratio, Type of coupling and Block change response can be altered by the NC part program:

0: Coupling parameters can be altered by the NC part program via command `COUPDEF`

1: Coupling parameters cannot be altered by the NC part program. Attempts to make changes will be rejected with an alarm message.

Superimposed motion

In synchronous operation, the synchronous spindle copies the movement of the leading spindle in accordance with the programmed transformation ratio.

At the same time, the synchronous spindle can also be traversed with overlay so that the LS and FS can operate at a specific angular position in relation to one another.

The overlaid traversing movement of the FS can be initiated in various ways:

- Programmable position offset of FS for AUTOMATIC and MDA:
  - The `COUPON` and `SPOS` language commands can be used for active synchronous operation to change the position reference between FS and LS (see Section "Selecting synchronous mode for a part program (Page 712)").
- Manual position offset of FS:
  - In JOG (JOG continuous or JOG incremental) mode

    Superimposition of FS using the handwheel or with plus or minus traversing keys when synchronous operation is active.
  - In AUTOMATIC and MDA modes

    Superimposition of FS with handwheel using DRF offset

As soon as the FS executes the overlaid traversing movement, IS "Overlaid movement" (DB31, ... DBX98.4) is set to the 1 signal.

The overlaid movement is executed optimally in terms of time at the maximum possible FS speed with `COUPON`. With an offset change by means of SPOS, the positioning velocity can be specified with FA[Sn] and manipulated by an override (can be selected through IS "Feedrate override valid for spindle" DB31, ... DBX17.0).

Note

For more information about specifying the position speed with FA[Sn]:

References:

Function Manual, Basic Functions; Spindles (S1), "Spindle modes, positioning operations" section

---

Setpoint correction

The setpoint correction of the system variable $AA_COUP_CORR[Sn] impacts on all subsequent following spindle programming in the same way as a position offset and corresponds to a DRF offset in the MCS.

Example: establish correction value

If a coupling offset of 7° has been programmed using COUPON(....,77) and if a mechanical offset of 81° has come about as a result of closing the workpiece support fixture, a correction value of 4° is calculated:

The system variables return the following values for the following spindle:

$P_COUP_OFFS[S2] ; programmed position offset = 77°

$AA_COUP_OFFS[S2] ; setpoint position offset = 77°

$VA_COUP_OFFS[S2] ; actual value position offset approx. 77°

$AA_COUP_CORR[S2] ; correction value = 4°

13.1.3 Prerequisites for synchronous mode

Conditions on selection of synchronous mode

The following conditions must be fulfilled before the synchronous spindle coupling is activated or else alarm messages will be generated.

- The synchronous spindle coupling must have been defined beforehand (either permanently configured via machine data or according to user definition via part program using `COUPDEF`).
- The spindles to be coupled must be defined in the NC channel in which the coupling is activated.

  Channel-spec. MD20070 $MC_AXCONF_MACHAX_USED

  axis spec. MD35000 $MA_SPIND_ASSIGN_TO_MACHAX
- The following spindle must be assigned to the NC channel in which the coupling is activated.

  Default setting with axis-specific MD30550 AXCONF_ASSIGN_MASTER_CHAN
- The following applies to setpoint and actual value couplings (DV, AV):

  FS and LS must at least have a position measuring system for recording positions and position controls must be started up.

  Note

  When position control is activated, the maximum setpoint speed of the LS is automatically limited to 90% (control reserve) of the maximum speed. The limitation is signaled via IS "Setpoint speed limited" (DB31, ... DBX83.1).

References:

Function Manual, Basic Functions; Spindles (S1)

- The following applies to setpoint couplings (DV):

  To ensure more accurate synchronization characteristics, the LS should be in position control mode (language instruction `SPCON`) before the coupling is activated.
- Before selecting the synchronous mode, the gear stage necessary for FS and LS must be selected. In synchronous mode, gear stage changeover and therefore oscillation mode are not possible for FS and LS. Upon request, an alarm message is generated.

Cross-channel coupling

The LS can be located in any channel.

- The LS can be exchanged between channels by means of "Axis exchange".
- When several following spindles are coupled to one leading spindle, the dynamic response of the coupling is determined by the weakest response as a function of the coupling factor. The acceleration rate and maximum speed are reduced for the leading spindle to such a degree that none of the coupled following spindles can be overloaded.
- The following spindle is always located in the channel in which the coupling has been activated using `COUPON` or `COUPONC`.

13.1.4 Selecting synchronous mode for a part program

Activate coupling COUPON, COUPONC

Language command `COUPON` activates the coupling in the part program between the programmed spindles with the last valid parameters and thus also activates synchronous mode. This coupling may be a fixed configuration or user-defined. The leading spindle and/or following spindle may be at standstill or in motion at the instant of activation.

Certain conditions must be fulfilled before synchronous operation can be activated (see Section "Prerequisites for synchronous mode (Page 711)").

The `COUPONC` command adopts the previous programmed direction of spindle rotation and spindle speed for the following and leading spindle in the part program. It is not possible to specify an angular offset.

COUPON activation variants

Two different methods can be selected to activate synchronous mode:

1. Fastest possible activation of coupling with **any angular reference** between leading and following spindles.

   COUPON(FS, LS)
2. Activation of coupling with a **defined angular offset** POS_{FS} between leading and following spindle. With this method, the angular offset must be programmed on selection.

   COUPON(FS, LS, POS_{FS})

Block change behavior

Before synchronous operation is selected, it must be determined under what conditions the block change must occur when synchronous mode is activated (see Section "Definition (COUPDEF) (Page 721)").

Determine current coupling status

The $AA_COUP_ACT[<axial expression>] axial system variable can be used in the NC part program to specify the current coupling status for the specified axis/spindle (see Section "Axial system variables for synchronous spindle (Page 726)"). As soon as the synchronous spindle coupling is active for the following spindle, bit 2 must be "1" when read.

Change defined angular offset

Language commands `COUPON` and `SPOS` allow the defined angular offset to be changed while synchronous mode is active. The following spindle is positioned as an overlaid movement at the angular offset programmed with POS_{FS}. During this time, the IS "overlaid movement" (DB31, ... DBX98.4) is set.

Angular offset POS_{FS}

The defined angular offset POS_{FS} must be specified as an absolute position referred to the zero degrees position of the leading spindle in a positive direction of rotation.

The "0° position" of a position-controlled spindle is calculated as follows:

- From the zero mark or Bero signal of the measurement system and
- From the reference values saved using axis-specific machine data:

  MD34100 $MA_REFP_SET_POS, reference point value,

  of no significance with interval-coded systems.

  MD34080 $MA_REFP_MOVE_DIST reference point distance/target point

  with interval-coded systems,

  MD34090 $MA_REFP_MOVE_DIST_CORR reference point offset / absolute offset with interval coding.

Range of POS_{FS}: 0 ... 359,999°.

References:

Function Manual Basic Functions; Reference Point Travel (R1)

Read current angular offset

Using axial system variables, it is possible to read the current position offset between the FS and LS in the NC part program. A distinction is made between:

- Current position offset of setpoint between FS and LS

  $AA_COUP_OFFS [<axis name for FS>]
- Current position offset of actual value between FS and LS

  $VA_COUP_OFFS [<axis name for FS>]

(Explanation of <axis name>, see Section "Synchronous mode (Page 705)")

Activation after power ON

Synchronous mode can also be activated with non-referenced/synchronized FS or LS (IS "Referenced/synchronized 1 or 2" DB31, ... DBX60.4 or DBX60.5 = 0). In this case, a warning message is displayed.

Example:

LS and FS are already coupled in a friction lock via a workpiece after power ON.

13.1.5 Deselecting the synchronous mode for the part program

Open coupling (COUPOF, COUPOFS)

Synchronous mode between the specified spindles is canceled by the parts program instruction `COUPOF`. Three variants are possible.

If synchronous mode is canceled between the specified spindles using `COUPOF`, then it is irrelevant whether this coupling is permanently configured or user defined. The leading and following spindles can be at standstill or in motion when synchronous operation is deactivated.

On switching off the synchronous mode with `COUPOF`, the following spindle is put into **control mode**. The originally programmed S-word is no longer valid for the FS, the following spindle can be operated like any other normal spindle.

When the coupling is opened with `COUPOF`, a block preprocessing stop `STOPRE` is generally initiated internally in the control.

The `COUPOFS` instruction can be used to open a coupling either as quickly as possible with a stop and no position data or with a stop at the programmed position.

COUPOF variants

Three different methods can be used to deselect synchronous mode with `COUPOF`:

1. Deactivation of coupling as quickly as possible

   The block change is enabled immediately.

   COUPOF(FS, LS)

2. A coupling is not deselected until the following spindle has crossed the programmed deactivation position POS_{FS}.

   The block change is then enabled.

   COUPOF(FS, LS, POS_{FS})

3. A coupling is not deselected until the following spindle and leading spindle has crossed the programmed deactivation positions POS_{FS} and POS_{LS}.

   The block change is then enabled.

   COUPOF(FS, LS, POS_{FS} , POS_{LS})

POS_{FS}, POS_{LS}

Deactivation positions POS_{FS} and POS_{LS} match the actual positions of FS and LS respectively referred to the defined reference point value.

Range of POS_{FS}, POS_{LS}: 0 ... 359,999°.

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

COUPOF during the motion

If synchronous mode is deselected while the spindles are in motion with `COUPOF`, the following spindle continues to rotate at the current speed (n_{FS}). The current speed can be read with system variable $AA_S in the NC parts program.

The following spindle can then be stopped from the parts program with M05, `SPOS`, `SPOSA` or from the PLC with the appropriate interface signal.

COUPOFS with stop of following spindle

Opening a synchronous spindle coupling is extended by a stop of the following spindle:

- Deactivating a coupling as quickly as possible and opening a coupling as quickly as possible.

  The block change is then enabled.

  COUPOFS(FS, LS)

- Opening the coupling with stop of following spindle at the programmed position. The block change is then enabled.

Condition:

COUPOFS(FS, LS) and COUPOFS(FS, LS, POS_{FS}) have no meaning if a coupling was active.

13.1.6 Controlling synchronous spindle coupling via PLC

Controlling following spindle via PLC

Using the coupling-specific, axial VDI interface signals, it is possible to control synchronization motions for the following spindle from the PLC program. This offers the option of utilizing the PLC to disable, suppress or restore a synchronization motion for the following spindle specified by offset programming.

These signals have no effect on the leading spindle. The following coupling-specific VDI signal (PLC → NCK) is available:

IS "Disable synchronization" (DB31, ... DBX31.5)

"Disable synchronization"

The synchronization motion for the following spindle is suppressed using the axial signal IS "Disable synchronization" (DB31, ... DBX31.5).

When the main run advances to a block containing part program instruction `COUPON` (FS, LS, offset), the following interface signal is evaluated for the following spindle:

IS "Disable synchronization" (DB31, ... DBX31.5).

- For IS "Disable synchronization" (DB31, ... DBX31.5) = 0, the position offset is traversed through as before.
- For IS "Disable synchronization" (DB31, ... DBX31.5) = 1, only the continuous velocity synchronism is established. The following spindle does not execute any additional movement.

  The coupling then responds analogously to a programmed `COUPON(<FS>,<LS>)`.

Special features

For the IS "Disable synchronization" (DB31, ... DBX31.5) offset motion of the following spindle cannot be controlled that was generated as follows:

- SPOS, POS
- Synchronized actions
- FC18 (for 840D sl)
- JOG

These functions are controlled by VDI signal IS "Feedrate stop/Spindle stop" (DB31, ... DBX4.3).

Synchronized state reached

Whenever a state of synchronism has been reached, the following two VDI signals are set regardless of whether synchronization has been disabled or not:

IS "Synchronism coarse" (DB31, ... DBX98.1) and

IS "Synchronism fine" (DB31, ... DBX98.0)

Further block changes after COUPON are not prevented by suppression of synchronization.

Example

Block change behavior after COUPON

Program code	Comment
	; IS "Disable synchronization"
	; set (DB31, ... DBX31.5) = 1 for S2
N51 SPOS=10 SPOS[2]=10	; Positions correspond to an offset
	; of 0°
N52 COUPDEF(S2,S1,1,1,"FINE","DV")	
N53 COUPON(S2,S1,77)	; Actual offset of 0 degrees is retained
	; no following spindle movement,
	; VDI signals
	; IS "Synchronism coarse"
	; (DB31, ... DBX98.1) and
	; IS "Synchronism fine"
	; (DB31, ... DBX98.0)
	; are set and the block change
	; enabled.
N54 M0	
N57 COUPOF(S2,S1)	
N99 M30	

Reset and recovery

Resetting the IS "Disable synchronization" (DB31, ... DBX31.5) has no effect on the following spindle offset. If the offset motion of the following spindle has been suppressed by the VDI interface signal, then the offset is not automatically applied when the VDI signal is reset.

Synchronization is recovered as follows:

- By repeating the part program operation COUPON (FS, LS, offset) with IS "Disable synchronization" (DB31, ... DBX31.5) = 0.

  COUPON (FS, LS, offset) can be written e.g. in an ASUB.
- By setting the IS "Resynchronize" (DB31, ... DBX31.4) = 1

Read offset

The following system variables can be used to read three different position offset values of the following spindle from the part program and synchronized actions. The variable $P_COUP_OFFS[Sn] is only available in the part program.

Description	NCK variable
Programmed position offset of the synchronous spindle	$P_COUP_OFFS[Sn]
Position offset of synchronous spindle, setpoint end	$AA_COUP_OFFS[Sn]
Position offset of synchronous spindle, actual value end	$VA_COUP_OFFS[Sn]

"Feedrate stop/spindle stop"

By configuring bit 4 in MD30455 MISC_FUNCTION_MASK, the behavior of the axial IS **"Feedrate stop/Spindle stop" (DB31, ... DBX4.3)** is defined for the following spindle.

Bit 4 = 0 compatibility method:

Canceling feed enable for the following spindle decelerates the coupling assembly.

Bit 4 = 1:

Feedrate enable refers only to the interpolation component (SPOS),..) and does not affect the coupling.

Note

Other configuration options for axis functions using MD30455 $MA_MISC_FUNCTION_MASK:

References:

Function Manual, Basic Functions; Rotary Axes (R2), Section: Programming rotary axes

---

13.1.7 Monitoring of synchronous operation

Fine/coarse synchronism

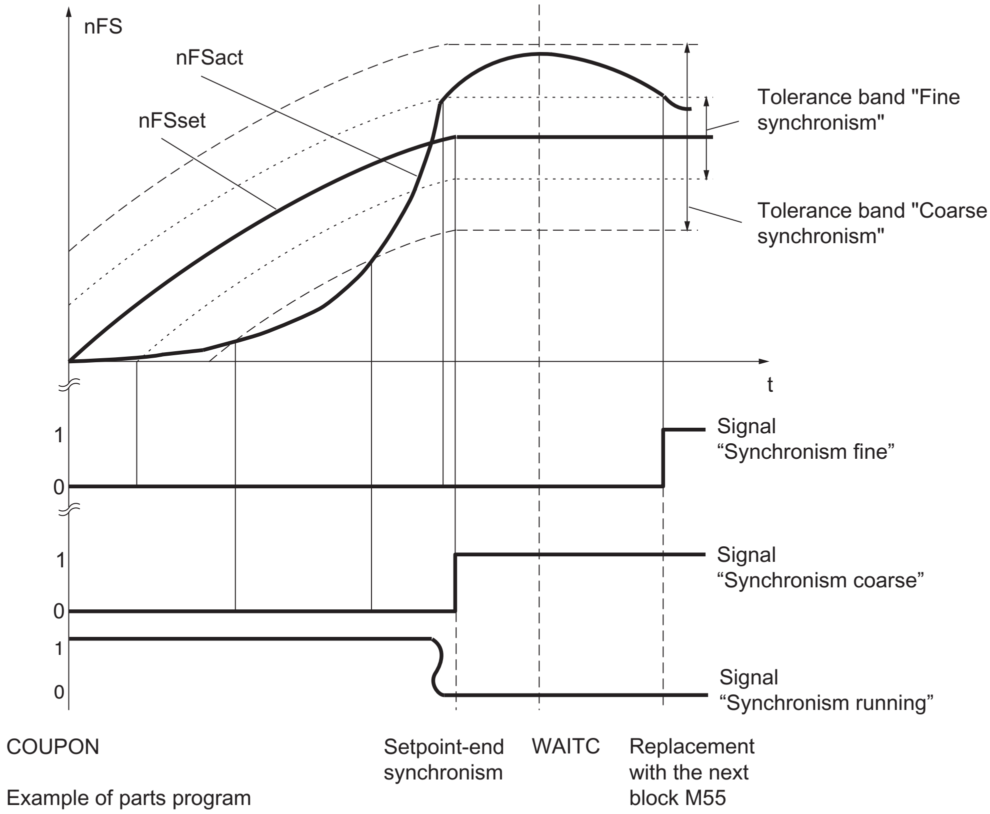
In addition to conventional spindle monitoring operations, synchronous operation between the FS and LS is also monitored in synchronous mode.

IS "Fine synchronism" (DB31, ... DBX98.0) or IS "Coarse synchronism" (DB31, ... DBX98.1) is transmitted to the PLC to indicate whether the current position (AV, DV) or actual speed (VV) of the following spindle lies within the specified tolerance window.

When the coupling is switched on, the signals "Coarse synchronism" and "Fine synchronism" are updated when setpoint synchronism is reached.

The size of the tolerance windows is set with machine data of the FS. Reaching of the synchronism is influenced by the following factors:

- AV, DV: Position variance between FS and LS
- VV: Difference in speed between FS and LS



Example of parts program

```
⋮
M3 S500 M2=5
G4 P10
COUPON (S2, S1)
Z10
WAITC(S2, “Fine”)
M55
⋮
```

nFSact = actual speed of following spindle
nFSset = setpoint speed of following spindle
nFS = speed of following spindle

Figure 13-3 Synchronism monitoring with `COUPON` and synchronism test mark `WAITC` with synchronization on a turning leading spindle

Threshold values

The relevant position or velocity tolerance range for the following spindle in relation to the leading spindle must be specified in degrees or 1 rev/min.

- Threshold value for "Coarse synchronism"

  axis spec. MD37200: AV, DV: COUPLE_POS_TOL_COARSE

  MD37220: VV: COUPLE_VELO_TOL_COARSE
- Threshold value for "Fine synchronism"

  axis spec. MD37210: AV,DV: COUPLE_POS_TOL_FINE

  MD37230: VV: COUPLE_VELO_TOL_FINE

Speed/acceleration limits

In synchronous mode, the speed and acceleration limit values of the leading spindle are adjusted internally in the control in such a way that the following spindle can imitate its movement, allowing for the currently selected gear stage and effective speed ratio, without violating its own limit values.

For example, the LS is automatically decelerated to prevent the FS from exceeding the maximum speed in order to maintain synchronism between the spindles.

13.2 Programming

Coupling commands with leading-spindle programming

Command	Function
`COUPDEF(FS, LS, ...)`	Define coupling or change for configured coupling
`COUPON(FS, LS, POSFS)`	Switch the coupling on
`COUPONC(FS, LS)`	Activate coupling with transfer of the currently effective speed of the following spindle
`COUPOF(FS, LS, POSFS, POSLS)`	Switch the coupling off
`COUPOFS(FS, LS, POSFS)`	Deactivate coupling with stop of the following spindle
`COUPDEL (FS, LS)`	Delete coupling
`COUPRES (FS, LS)`	Reactivate configured coupling data

Coupling commands without leading-spindle programming

Command	Function
`COUPOF(FS, POSFS, POSLS)`	Switch the coupling off
`COUPOFS(FS, POSFS)`	Deactivate coupling with stop of the following spindle
`COUPDEL(FS)`	Delete coupling
`COUPRES(FS)`	Reactivate configured coupling data

See also

Definition (COUPDEF) (Page 721)

Switch the coupling (COUPON, COUPONC, COUPOF) on and off (Page 724)

13.2.1 Definition (COUPDEF)

Programmable couplings

The number of couplings can be programmed as often as desired depending on the axes available. This number results from the number of axes/spindles less one for the master. Furthermore, one coupling can also be configured via machine data as in earlier SW versions.

Permanently configured coupling

The coupling characteristics and transformation ratio for a permanently configured synchronous spindle coupling can be altered by the NC part program provided that they are not write-protected. The machine axes for LS and FS cannot be changed.

Define new couplings

Language command "`COUPDEF`" can be used to create new synchronous spindle couplings (user-defined) and to modify the parameters for existing couplings.

When the coupling parameters are fully specified, the following applies:

COUPDEF(FS, LS, $T_{numerator}$, $T_{denominator}$, block change behavior, coupling type)

The synchronous spindle coupling is unambiguously defined with FS and LS

The other coupling parameters must only be programmed when they need to be changed. The last valid status remains applicable for non-specified parameters.

The individual coupling parameters are explained below:

- **FS, LS**: Spindle name for following and leading spindles

  e.g.: S1, SPI(1), S2, SPI(2)

  The valid spindle number must be assigned in the axis-specific MD35000 $MA_SPIND_ASSIGN_TO_MACHAX of a machine axis.

- **$T_{numerator}$, $T_{denominator}$**: Transformation ratio parameters for numerator and denominator

  The transformation ratio is specified in the form of numeric values for numerator and denominator (see Section "Synchronous mode (Page 705)").

  The numerator must always be programmed. If no denominator is specified, then its value is always assumed to be "1.0".

- Block change behavior

  This parameter allows you to select when the block change should take place when synchronous operation is selected:

NOC: Block change is enabled immediately

FINE: Block change in response to "Fine synchronism"

COARSE: Block change in response to "Coarse synchronism"

IPOSTOP: Block change for IPOSTOP (i.e. after setpoint-end synchronism)

The block change response is specified as a character string (i.e. with quotation marks).

The block change response can be specified simply by writing the letters in bold print. The remaining letters can be entered to improve legibility of the part program but they are not otherwise significant.

If no block change response is specified, then the currently selected response continues to apply.

With the programmable synchronism test markers WAITC, the replacement with new blocks is delayed until the parameterized synchronism is reached.

- Coupling type

DV (Desired Values): Setpoint coupling between FS and LS

AV (Actual Values): Actual value coupling between FS and LS

VV (Velocity Values): Speed coupling between FS and LS

If no coupling type is specified, then the currently selected type continues to apply.

Note

The coupling type may only be changed when synchronous operation is deactivated!

Examples

COUPDEF (SPI(2), SPI(1), 1.0, 1.0, "FINE", "DV")

COUPDEF (S2, S1, 1.0, 4.0)

COUPDEF (S2, SPI(1), 1.0)

Default settings

The following default settings apply to user-defined couplings:

- $Ü_{Numerator}$=1.0
- $Ü_{Denominator}$ =1.0
- Block change response = **IPOSTOP** (block change enabled with setpoint synchronism)
- Type of coupling = **DV** (setpoint coupling)

Delete couplings

Language command "COUPDEL" is used to delete user-defined couplings.

COUPDEL (FS, LS)

Note

COUPDEL impacts on an active coupling, deactivates it and deletes the coupling data. Alarm 16797 is therefore meaningless.

The following spindle adopts the last speed. This corresponds to the behavior associated with COUPOF(FS, LS).

Activate original coupling parameters

Language command "COUPRES" can be used to re-activate the configured coupling parameters.

COUPRES (FS, LS)

The parameters modified using COUPDEF (including the transformation ratio) are subsequently deleted.

Language command "COUPRES" activates the parameters stored in the machine and setting data (configured coupling) and activates the default settings (user-defined coupling).

Programmable block change

It is possible to mark a point in the NC program using the "WAITC" language command. The system waits at this point for fulfillment of the synchronism conditions for the specified FS and delays changes to new blocks until the specified state of synchronism is reached (see "Figure 13-3 Synchronism monitoring with COUPON and synchronism test mark WAITC with synchronization on a turning leading spindle (Page 719)").

WAITC (FS)

Advantage: The time between activating the synchronous coupling and reaching synchronism can be used in a meaningful way, technologically speaking.

Note

Basically, it is always possible to write WAITC. If the spindle indicated is not active as FS, the command for this spindle is without effect.

If no synchronism condition is indicated, the check is always performed for the synchronism condition programmed/configured on the respective coupling, at least for the setpoint synchronism.

Examples:

WAITC(S2),
:
WAITC(S2, "Fine"),
:
WAITC(S2, ,S4, "Fine")

Stop and block change

If "Stop" has been activated for the cancellation period of the axis enables for the leading or following spindle, then the **last** setpoint positions with the setting of the axis enables from the servo drive are approached again.

Commands COUPON and WAITC can influence the block change behavior. The block change criterion is defined using COUPDEF or via the MD21320 $MC_COUPLE_BLOCK_CHANGE_CTRL_1.

13.2.2 Switch the coupling (COUPON, COUPONC, COUPOF) on and off

Activate synchronous mode

Language command COUPON is used to activate couplings and synchronous mode.

Two methods by which synchronous operation can be activated are available:

1. **COUPON(FS, LS)**

   Fastest possible activation of synchronous operation with any angular reference between the leading and following spindles.

2. **COUPON(FS, LS, POS_{FS})**

   Activation of synchronous operation with a defined angular offset POS_{FS} between the leading and following spindles. This offset is referred to the zero degrees position of the leading spindle in a positive direction of rotation. The block change is enabled according to the defined setting. Range of POS_{FS}: 0 ... 359.999 degrees.

3. **COUPONC(FS, LS)**

   When activating with COUPONC, the previous programming of M3 S... or M4 S... is adopted. A difference in speed is transferred immediately. An offset position cannot be programmed.

By programming COUPON(FS, LS, POS_{FS}) or SPOS when synchronous operation is already active, the angular offset between LS and FS can be changed.

Deactivate synchronous mode

Three different methods can be selected to deactivate synchronous mode:

1. **COUPOF(FS, LS)**

   Fastest possible deactivation of synchronous operation. The block change is enabled immediately.

2. **COUPOF(FS, LS, POS_{FS})**

   Deselection of synchronous operation after deactivation position POS_{FS} has been crossed. Block change is not enabled until this position has been crossed.

3. **COUPOF(FS, LS, POS_{FS}, POS_{LS})**

   Deselection of synchronous operation after the two deactivation positions POS_{FS} and POS_{LS} have been crossed. Block change is not enabled until **both** programmed positions have been crossed.

   Range of POS_{FS}, POS_{LS}: 0 ... 359,999°.

If continuous path control (G64) is programmed, a non-modal stop is generated internally in the control.

Examples:

```
COUPDEF (S2, S1, 1.0, 1.0, "FINE, "DV")
:
COUPON (S2, S1, 150)
:
COUPOF (S2, S1, 0)
:
COUPDEL (S2, S1)
```

1. **COUPOFS(FS, LS)**

   Deactivating a coupling with stop of the following spindle. Block change is performed as quickly as possible with immediate block change.

2. **COUPOFS(FS, LS, POS_{FS})**

   After the programmed deactivation position that refers to the machine coordinate system has been crossed, the block change is not enabled until the deactivation positions POSFS have been crossed.

   Value range 0 ... 359.999°.

13.2.3 Axial system variables for synchronous spindle

Determining current coupling status

The current coupling status of the following spindle can be read in the NC part program with the following axial system variable:

$AA_COUP_ACT[<axial expression>]

For explanation of <axial expression>, see Section "Synchronous mode (Page 705)".

Example:

$AA_COUP_ACT[S2]

The value read has the following significance for the following spindle:

Byte = 0:	No coupling active
Bit 2 = 1:	Synchronous spindle coupling active
Bit 2 = 0:	Synchronized spindle coupling is not active

Read current angular offset

The current position offset between the FS and LS can be read in the NC part program by means of the following axial system variables:

- Setpoint-based position offset between FS and LS:

  $AA_COUP_OFFS[<axial expression>]
- Actual-value-based position offset between FS and LS:

  $VA_COUP_OFFS[<axial expression>]

Example:

$AA_COUP_OFFS[S2]

If an angular offset is programmed with `COUPON`, this coincides with the value read after reading the setpoint synchronization.

Reading the programmed angular offset

The position offset last programmed between the FS and LS can be read in the NC part program by means of the following axial system variables:

$P_COUP_OFFS[<axial expression>]

Note

After cancellation of the servo enable signal when synchronous operation and follow-up mode are active, the position offset applied when the controller is enabled again is different to the originally programmed value.

$P_COUP_OFFS only returns the value originally programmed. $AA_COUP_OFFS and $VA_COUP_OFFS return the current value. The programmed offset can be recreated with NST DB31, ... DBX31.4 (resynchronization).

13.2.4 Automatic selection and deselection of position control

Behavior in speed control mode

In DV coupling mode, program instructions COUPON, COUPONC and COUPOF, COUPOFS are used to activate and/or deactivate position control for the leading spindle as required. If there are several following spindles on the leading spindle, then in speed-controlled mode, the **first** DV **activates** coupling position control for the leading spindle and the **last** DV coupling **deactivates** coupling position control for the leading spindle if SPCON is not programmed.

The leading spindle does not need to be located in the same channel as the following spindle.

Automatic selection with COUPON and COUPONC

Depending on the coupling type, the effect of COUPON and COUPONC on the position control for synchronous operation is as follows:

Coupling type	DV	AV	VV
Following spindle FS	Position control ON	Position control ON	No action
Leading spindle LS	Position control On [1]	No action	No action

[1] The position control is activated by a COUPON and COUPONC instruction if **at least one** following spindle has been coupled to it with coupling type DV.

Automatic deselection with COUPOF and COUPOFS

Depending on the coupling type, the effect of COUPOF and COUPOFS on the position control is as follows:

Coupling type	DV	AV	VV
Following spindle FS	Position control OFF [2]	Position control OFF [2]	No action [2]
Leading spindle LS	Position control OFF [3]	No action	No action

[2] COUPOF and COUPOFS without position specification

Speed control mode is activated for the following spindle. Positioning mode is activated with COUPFS with a stop position. Position control is **not deactivated** if the following spindle was located in position-controlled spindle mode using SPCON or COUPFS was programmed with position.

[3] With COUPOF position control is **deactivated** if there are no more couplings of the DV coupling type for this leading spindle. Position control **is retained** if the leading spindle is in positioning mode or axial mode **or** was in position-controlled spindle mode using SPCON.

13.3 Configuration

> **Note**
>
> **One** synchronous-spindle coupling can be configured for each channel.

Table 13- 1 Machine data

Number	Name: $MC_	Function
MD21300	COUPLE_AXIS_1[<n>]	Machine axes of the synchronous-spindle coupling: • <n> = 0: Machine axis number of the following spindle • <n> = 1: Machine axis number of the leading spindle Machine axis numbers in accordance with: MD20070 $MC_AXCONF_MACHAX_USED (machine axes in the channel) Machine axis numbers == 0: No coupling configured. The following system data is then not relevant. **Note**: The machine axes configured for the synchronous-spindle coupling **cannot** be changed using program commands.
MD21320	COUPLE_BLOCK_ CHANGE_CTRL_1	Block change release after activating the synchronous operation 1): • Immediately • On reaching "Synchronism fine" • On reaching "Synchronism coarse" • On reaching "Synchronism setpoint" **Note:** No change protection1), the block change release can be changed with the `COUPDEF` command.
MD21310	COUPLING_MODE_1	Coupling type 1): • Actual value coupling • Setpoint value coupling • Speed coupling **Note:** No change protection1), the coupling type can be changed for **deactivated** coupling with the `COUPDEF` command.
MD21330	COUPLE_RESET_ MODE_1	Behavior of the synchronous-spindle coupling with regard to NC Start, NC Stop and Reset.
MD21340	COUPLE_IS_WRITE_ PROT_1	Change protection for coupling
1) See MD21340 $MC_COUPLE_IS_WRITE_PROT_1		

Table 13- 2 Setting data

Number	Name: $SC_	Function
SD42300	COUPLE_RATIO_1[<n>]	Speed transmission ratio: FS / LS = numerator / denominator 1): • <n> = 0: Numerator (FS) • <n> = 1: Denominator (LS) **Note:** No change protection1), the transmission ratio can be changed with the `COUPDEF` command.
1) See MD21340 $MC_COUPLE_IS_WRITE_PROT_1		

13.3.1 Response of the synchronous-spindle coupling for NC Start

The behavior of the synchronous-spindle coupling during NC Start depends on the setting in the following machine data:

Configured synchronous-spindle coupling

Response	MD21330 $MC_COUPLE_RESET_MODE_1
Maintain coupling	Bit 0 = 0
Deselect coupling	Bit 0 = 1
Activate configured data	Bit 5 = 1
Switch the coupling on	Bit 9 = 1

Programmed synchronous-spindle coupling

Response	MD20112 $MC_START_MODE_MASK
Maintain coupling	Bit 10 = 0
Deselect coupling	Bit 10 = 1

13.3.2 Behavior of the synchronous-spindle coupling for reset

The behavior of the synchronous operation for reset and at program end depends on the setting in the following machine data:

Configured synchronous-spindle coupling

Response	MD21330 $MC_COUPLE_RESET_MODE_1	MD20110 $MC_RESET_MODE_MASK
Maintain coupling	Bit 1 = 0	Bit 0 = 1
Deselect coupling	Bit 1 = 1	Bit 0 = 1
Activate configured data	Bit 6 = 1	Bit 0 = 1

Programmed synchronous-spindle coupling

	MD20110 $MC_RESET_MODE_MASK
Maintain coupling	Bit 0 = 1, Bit 10 = 1
Deselect coupling	Bit 0 = 1, Bit 10 = 0

13.4 Points to note

13.4.1 Special features of synchronous mode in general

Control dynamics

When using the setpoint coupling, the position control parameters of FS and LS (e.g. K_V factor) should be matched with one another. If necessary, different parameter blocks should be activated for speed control and synchronized mode. The control parameters of the following spindle may differ from position control, feedforward control and parameter block, as also in the uncoupled case, set using MD30455 $MA_MISC_FUNCTION_MASK (see Section "Special points regarding start-up of a synchronous spindle coupling (Page 743)").

Precontrol

Due to the improved control system dynamic response it provides, feedforward control for the following and leading spindles in synchronous mode is **always active.**

It can, however, be deselected for FS and LS with axis-specific MD32620 $MA_FFW_MODE. If MD32620 $MA_FFW_MODE is set to zero, there are function limitations. Position control can no longer be activated in motion with SPCON. SPOS, M19 or SPOSA are therefore not possible. The NC part program cannot deactivate the feedforward control for LS and FS with FFWOF.

The feedforward control mode (speed or torque feedforward control) is defined in axis-specific MD32620 $MA_FFW_MODE (see also Section "K3: Compensations (Page 223)").

Speed and acceleration limits

The speed and acceleration limits of the spindles operating in synchronous mode are determined by the "weakest" spindle in the synchronous spindle pair. The current gear stages, the programmed acceleration and, for the leading spindle, the effective position control status (On/Off) are taken into account for this purpose.

The maximum speed of the leading spindle is calculated internally in the control taking into account the speed ratio and the spindle limitations of the following spindle.

Multiple couplings

If the system detects that a coupling is already active for an FS and LS when the synchronous mode is activated, then the activation process is ignored and an alarm message is generated.

Examples of multiple couplings:

- A spindle is acting as the FS for several LS.
- Coupling cascade (an FS is an LS of an additional coupling)

Number of configurable spindles per channel

Every axis in the channel can be configured as a spindle. The number of axes per channel depends on the control version.

Cross-channel setpoint coupling

- Cross-channel synchronous spindle couplings can be implemented with no additional restrictions for DV, AV, and VV.
- Any number of following spindles, corresponding to the number of all spindles minus one spindle for the master, in any channels on an NCU can be coupled to one leading spindle.

Start synchronous mode using ASUB

When the PLC starts an ASUB, in the AUTOMATIC or MDA modes, synchronous operation can be switched on and off – or terminated.

References:

Function Manual, Basic Functions, Mode Group, Channel, Program Operation, Reset Behavior (K1)

Response to alarms

In the case of an alarm, which occurs during synchronous operation, and acts as alarm response "withdraw control enable" and "activate follow-up mode" in the control, the ongoing control behavior is the same as the behavior due to NC/PLC interface signals:

- DB31, ... DBX2.1 = 0 (controller enable)
- DB31, ... DBX1.4 = 1 (follow-up mode)

(See Section "Synchronous mode and NC/PLC interface signals (Page 734)")

By resynchronizing via the NC/PLC interface signal:

DB31, ... DBX31.4 = 0 → 1 (resynchronization)

If the programmed offset is restored (see Section "Restore synchronism of following spindle (Page 732)").

Block search when synchronous operation is active

Note

When synchronous operation is active for a block search, then it is recommended that only block search type 5, "Block search via program test" (SERUPRO), is used.

13.4.2 Restore synchronism of following spindle

Causes for a positional offset

When the coupling is reactivated after the drive enable signals have been canceled, a positional offset can occur between the leading and following spindles if follow-up mode is activated. A positional offset can be caused by:

- A part has been clamped or both spindles have been turned manually (machine area is open, drives are disconnected from supply).
- After the spindle enable signals are canceled, the two spindles coast to standstill at different speeds if they are not mechanically coupled.
- A drive alarm occurs (internal follow-up mode):

  DB31, ... DBX61.3 (follow-up mode active) = 1

  When the alarm is cleared, the NC must not trigger any synchronization motion.
- A synchronization was not executed due to a synchronization lock of the following spindle:

  DB31, ... DBX29.5 (Disable synchronization)

Basic procedure

If the following and leading spindles have fallen out of synchronism, or failed to synchronize at all, synchronism can be restored between them by the following measures:

1. Set the axis enable signals and cancel synchronization disable signal if this has been set.
2. Start following spindle resynchronization with the NC/PLC interface signal:

   DB31, ... DBX31.4 (resynchronization)

   Only after the resynchronization process is complete can the setpoint-end synchronism be fully restored.
3. Wait until the coupled spindles have synchronized.

Enable resynchronization

Setting the enabling signals closes the coupling at the current actual positions. The two following NC/PLC interface signals are set:

DB31, ... DBX98.1 (coarse synchronism)

DB31, ... DBX98.0 (fine synchronism)

The following **requirements** must be fulfilled for resynchronization to work:

- The axis enabling signal must be set for the following spindle.
- The PLC must not set any synchronization disables for the following spindle:

  DB31, ... DBX31.5 (Disable synchronization)

Resynchronize following spindle

Resynchronization is started for the relevant following spindle and commences as soon as the low-high edge of following interface signal is detected:

DB31, ... DBX31.4 (resynchronization)

The NC acknowledges the detection of the edge by outputting the NC/PLC interface signal:

DB31, ... DBX99.4 (synchronization running)

The interface signal "Synchronization running" is reset if:

- synchronization of the following spindle has been completed up to the stage at which there is synchronism at the setpoint end.
- the NST DB31, ... DBX31.4 (resynchronization) was reset.

Response of synchronous signals during additional movements for the following spindle

The superimposed component is calculated to establish the synchronism signals.

Example

Program code	Comment
N51 SPOS=0 SPOS[2]=90	
N52 OUPDEF(S2,S1,1,1,"FINE","DV")	
N53 COUPON(S2,S1,77)	
N54 M0	; Offset=77°, "coarse", "fine" synchronous run signals exist.
N55 SPOS[2]=0 FA[S2]=3600	; Difference in speed, synchronism signals "coarse", "fine" are reported
N56 M0	; (note tolerances, see above)
	; Offset=0°, "coarse", "fine" synchronous run signals exist.
N60 M2=3 S2=500	; difference in speed, synchronism signals "coarse", "fine" are reported.
	; offset undefined, synchronism signals "coarse", "fine" are reported.
N65 M0	; (Note tolerances, see above)

Note

The axis enable signals can be canceled to interrupt a movement overlaid on the following spindle (e.g. SPOS). This component of the movement is not affected by IS "NC/PLC interface signal" DB31, ... DBX31.4 (resynchronization), but is restored by the REPOS operation.

Supplementary condition

IS DB31, ... DBX31.4 (resynchronization) has any effect only if there is a **defined offset position** between the following spindle and leading spindle.

This is the case following `COUPON` with offset positions such as `COUPON(...,77)` or `SPOS`, `SPOSA`, `M19` for the following spindle with a closed coupling..

13.4.3 Synchronous mode and NC/PLC interface signals

Note

During synchronous operation, the effect of the associated interface signal for the leading (LS) or following spindle (FS) on the coupling must always be considered.

Spindle override (DB31, ... DBB19)

Only the spindle compensation value of the LS acts in the synchronous operation.

Spindle disable (DB31, ... DBX1.3)

LS	FS	coupling	Response
0	0	off	Setpoints are output
0	1	off	No setpoint output for FS
1	0	off	No setpoint output for LS
1	1	off	No setpoint output for LS and FS
0	0	ON	Setpoints are output
0	1	ON	Spindle disable not effective for FS
1	0	ON	Spindle disable also effective for FS, no setpoint output
1	1	ON	No setpoint output for LS and FS

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

LS: Resetting the "controller enable" during synchronous operation

If the controller enable of the LS is reset during synchronous operation for active **setpoint** coupling, a control-internal switching is made to the **actual value** coupling. If the controller enable is reset while the LS is traversing, the LS is stopped and an alarm issued. Synchronous operation remains active.

LS and FS: Selection of synchronous operation without controller enable

If the "controller enable" for LS and FS is not set for selected synchronous operation, it will be activated. LS and FS, however, are not traversed until controller enable for LS **and** FS has been granted.

LS and FS: Setting the "Controller enable"

The position assumed by a spindle with setting the "controller enable" depends on DB31, ... DBX1.4 == <value> (follow-up operation):

<value>	Assumed spindle position
0	Position when controller enable is canceled
1	Current position

Note

It is recommended for synchronous spindles during a block search, to write the DB31, ... DBX2.1 interface signal (controller enable) always together for FS **and** LS. If this is not done, the block search, for example, stops after a FS machining because the controller enable of the LS is not pending.

Note

Synchronism error

If the DB31, ... DBX2.1 interface signal (controller enable) is canceled for the FS after Spindle Stop without the coupling being deactivated beforehand, then any synchronism error resulting from external intervention will not be compensated when the "controller enable" is activated again. This causes any programmed angular relationship between FS and LS to be lost.

The angular relationship can be restored by resynchronizing: DB31, ... DBX31.4 = 1 (resynchronization)

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

For follow-up operation, the set position is regularly set to the current actual position:

DB31, ... DBX1.4 == 1 (follow-up operation) **AND** DB31, ... DBX2.1 == 0 (controller enable)

⇒ Cyclic: Set position = actual position

Note

DB31, ... DBX1.4 (follow-up operation) is relevant only for DB31, ... DBX2.1 == 0 (controller enable)

Position measuring system 1/2 (DB31, ... DBX1.5 and 1.6)

Switchover of the position measuring system for the FS and LS is possible during synchronous operation. The coupling is retained.

Note

It is recommended to switchover the position measuring system for FS and LS only for deselected synchronous operation.

Delete distance-to-go / spindle reset (DB31, ... DBX2.2)

LS: Setting spindle reset during synchronous operation

The setting of spindle reset causes the LS to be braked to standstill with the parameterized acceleration. Synchronous operation remains active.

Any superimposed movement, other than together with COUPON / COUPONC, will be completed as fast as possible.

Spindle stop (feed stop) (DB31, ... DBX4.3)

LS and FS: Setting spindle stop during synchronous operation

The setting of "spindle stop" for FS or LS causes both spindles to be braked synchronous to standstill. Synchronous operation remains active.

Resetting spindle stop

Once "spindle stop" has been reset for both spindles, reacceleration is made to the last valid speed setpoint.

Application example

Bring FS and LS to a standstill when a protection door is opened during synchronous operation.

Signal characteristic for LS and FS:

1. Stop: DB31, ... DBX4.3 = 1 (spindle stop)
2. Waiting for standstill: DB31, ... DBX61.4 == 1
3. Stopping: DB31, ... DBX2.1 = 0 (controller enable)

Delete S value (DB31, ... DBX16.7)

LS: Delete S value during synchronous operation

If "delete S value" is set, the LS is braked to a standstill using a ramp. Synchronous operation remains active.

FS: Delete S value during synchronous operation

The control interface signal does not have any function for the FS in synchronous operation.

Resynchronize spindle 1/2 (DB31, ... DBX16.4 and 16.5)

LS: Resynchronizing the position measuring system during synchronous operation

---

Note

It is recommended to resynchronize the position measuring system of the LS only for deselected synchronous operation.

Resynchronize (DB31, ... DBX31.4)

FS: Restoring the programmed angular offset

If synchronism between FS and LS is lost or not performed, the programmed angular offset can be restored.

- Requirement: DB31, ... DBX31.4 = 1 (resynchronization)
- Acknowledgment: DB31, ... DBX99.4 == 1 (synchronization running)

Traverse keys for JOG (DB31, ... DBX4.6 and 4.7)

The "plus and minus traversing keys" for JOG are **not disabled** in the control for the FS in synchronous operation, i.e. the FS executes a superimposed motion if one of these keys is pressed.

Note

If superimposed traversing movements are to be precluded, they must be locked out by measures in the PLC user program.

NC Stop axes plus spindles (DB21, ... DBX7.4)

"NC Stop axes plus spindles" in synchronous operation decelerates the coupled spindles in accordance with the selected dynamic response. They continue to operate in synchronous mode.

NC Start (DB21, ... DBX7.1)

(See Section "Response of the synchronous-spindle coupling for NC Start (Page 729)")

Note

NC Start after NC Stop does not deselect synchronous operation.

13.4.4 Differential speed between leading and following spindles

When does a differential speed occur?

A differential speed develops, e.g. with turning machine applications, when two spindles are opposite each other. Through the signed addition of two speed sources, a speed component is derived from the leading spindle via the coupling factor. In addition to this, it is possible to program the following for the following spindle:

- speed with S... and
- direction of rotation with M3, M4 or M5

The spindles can normally be synchronized by a coupling factor with the value '-1'. This sign reversal then results in a differential speed for the following spindle as compared to an additional programmed speed. This typical behavior in relation to the NC is illustrated in the following diagram.

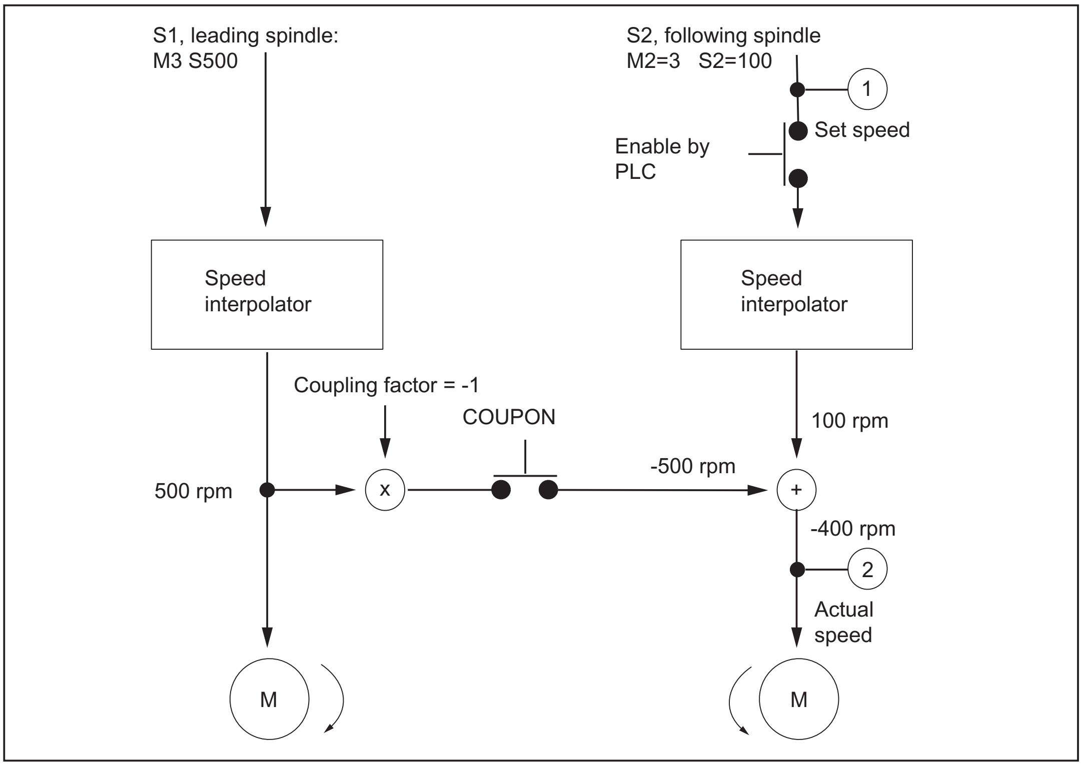


Figure 13-4 Schematic representation of process resulting in differential speed

Example

Program code	Comment
N01 M3 S500	; S1 rotates positively at 500 rpm
	; the master spindle is spindle 1
N02 M2=3 S2=300	, S2 rotates positively at 300 rpm

Program code	Comment
N05 G4 F1;	
N10 COUPDEF(S2,S1,-1)	; Coupling factor -1:1
N11 COUPON(S2,S1)	; Activate coupling, the speed of following spindle S2
	; results from the speed of the main spindle S1 and
	; the coupling factor
N26 M2=3 S2=100	; Programming of the speed difference,
	; S2 is the following spindle

Application

Manufacturing operations with positioned leading spindle and rotating tools require exact synchronism with the counter spindle which then functions like a following spindle. A turret rotating about the following spindle allows parts to be machined with different tool types. The following diagram shows an application in which the tool is positioned parallel to the main spindle.

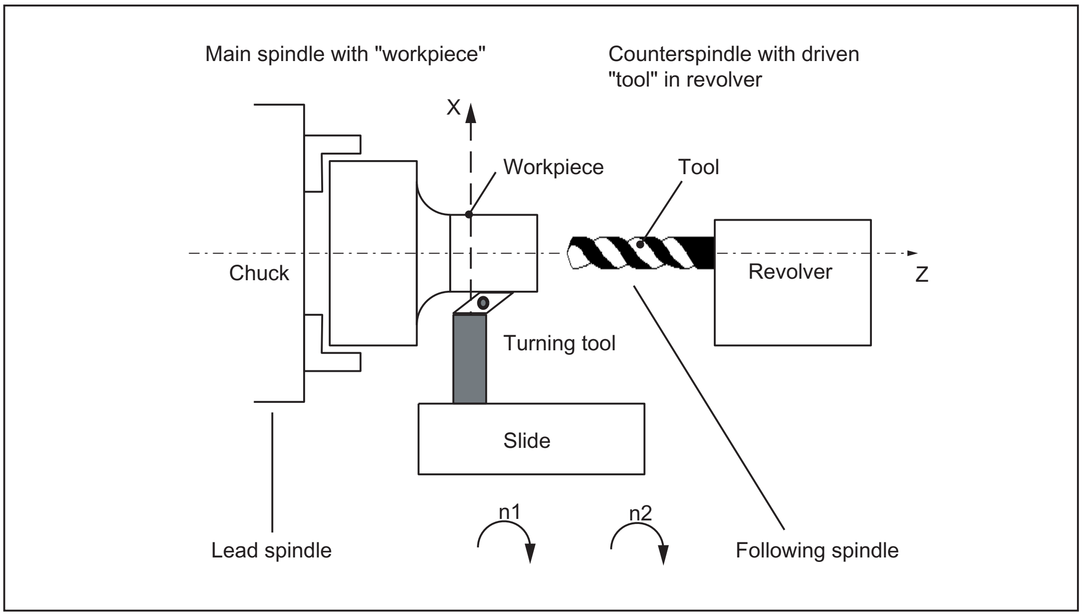


Figure 13-5 Application on a single-slide turning machine with turret about Z axis

Requirements

Basic requirements for differential speed programming:

- Synchronous spindle functionality is required.
- The dynamic response of the following spindle must be at least as high as that of the leading spindle. Otherwise, the system may suffer from reduced quality, for example, rigid tapping without a compensating chuck G331/G332.

- The differential speed must be programmed in the channel in which the following spindle is also configured. The leading spindle can be programmed in a different channel.
- The differential speed must be enabled for the following spindle by the PLC via IS "Enable overlaid movement" (DB31, ... DBX26.4). If the enable signal has not been set, alarm 16771 "Channel% Following axis% Overlaid movement not enabled" is output. This alarm is cleared when IS "Enable overlaid movement" (DB31, ... DBX26.4) is set or the coupling is terminated.

Note

The differential speed does not therefore affect the coupling process.

The following or leading spindle cannot change gear stages while a coupling is active.

Activate coupling with COUPONC

When the coupling is activated, the following spindle is accelerated, as before, to the leading spindle speed through application of the coupling factor. If the following spindle is already rotating (M3, M4) when the coupling is activated, it continues with this motion after coupling.

Deactivate coupling

If the coupling is deactivated, the following spindle continues to rotate at the speed corresponding to the sum of both speed components. The spindle behaves as if it had been programmed with the speed and direction transferred from the other spindle. When deactivating, there are no differences to the previous behavior.

Differential speed

A differential speed results from the **renewed** programming of the following spindle (in the example, S2=...) or M2=3, M2=4 in speed control mode **during** an active synchronous spindle coupling or by adopting the speed of the following spindle for COUONC.

Condition:

Speed S... must also be re-programmed with direction of rotation M3 or M4. Otherwise alarm 16111 "Channel% Block% Spindle% No speed programmed" is displayed.

Read offsets of following spindle

The current offset always changes when a differential speed is programmed. The current offset can be read at the setpoint end with $AA_COUP_OFFS[Sn] and at the actual value end with $VA_COUP_OFFS[Sn].

The last offset programmed returns the variable $P_COUP_OFFS[Sn].

Display differential speed

The programmed difference component is displayed as the speed setpoint for the programmed differential speed (in our example, corresponds to 100 rpm).

The actual speed refers to the motor speed. In the example, the actual speed is 500 rev/min * (-1) + 100 rpm = -400 rev/min.

IS NCK to PLC

Following spindle in speed-controlled operation

The IS "Spindle in setpoint range" (DB31, ... DBX83.5) is set for the following spindle by the NCK if the programmed speed difference (see previous example, N26 with M2=3 S2=100) is reached. If a differential speed is programmed and not enabled by the PLC, this VDI interface signal is not set.

Even if a differential speed has been programmed, the following spindle remains under position control if this is required by the coupling.

Note

The axial VDI interface signal NCK → PLC IS "Superimposed motion" (DB31, ... DBX98.4) is set when the differential speed programming creates setpoints in addition to the coupling setpoints.

Actual direction of rotation clockwise (DB31, ... DBX83.7)

IS "Actual direction of rotation clockwise" (DB31, ... DBX83.7) refers to the resulting motor direction.

IS PLC to NCK

Influence on following spindle via PLC interface

The effect of the axial VDI interface signals on the following spindle with differential speed in speed control mode is described below.

Delete distance to go / Spindle Reset (DB31, ... DBX2.2)

The programmed differential speed and direction can be terminated by IS "Delete distance-to-go / Spindle Reset" (DB31, ... DBX2.2). To delete the programmed speed only, it is possible to set IS "Delete S value" (DB31, ... DBX16.7).

Resynchronize spindle 1/2 (DB31, ... DBX16.4 and 16.5)

The IS "Resynchronize spindle 1/2" (DB31, ... DBX16.4/16.5) are **not** locked. Any positional offset is not compensated automatically by the coupling.

Invert M3/M4 (DB31, ... DBX17.6)

IS "Invert M3/M4" (DB31, ... DBX17.6) only inverts the additional programmed speed component for the following spindle.

The motion component generated by the synchronous spindle coupling remains unaffected.

Spindle override (DB31, ... DBB19)

The "Spindle override" VDI interface (DB31, ... DBB19) only affects the additional programmed speed component for the following spindle. If the spindle override switch is transferred to all axial inputs, then any change in the spindle override value is applied **doubly** to the following spindle:

- once indirectly by a change in speed for the leading spindle and
- once in the programmed component of the following spindle.

The offset value can be adjusted accordingly in the PLC program.

Coupling deselection

If the coupling is deactivated, the following spindle continues to rotate at the speed corresponding to the sum of both speed components. The motion transition upon coupling deselection is at continuous speed.

With `COUPOF`, the spindle behaves as if it had been programmed with the speed and direction transferred from the other spindle. In the example, this would be M4 S400.

When `COUPOFS` is programmed, the following spindle is decelerated to standstill from the current speed.

Activate additional functions

The following spindle can also be a master spindle. In this case, it is capable of additional functions.

- Rotational feedrate with G95, G96 and G97. With G96 S2=... the "constant cutting speed" can be activated for the following spindle.

  The speed dependent on the position of the transverse axis is the setpoint speed for the speed interpolator of spindle 2 and is therefore included in the total speed of S2.
- Rigid tapping without compensating chuck with G331, G332.

13.4.5 Behavior of synchronism signals during synchronism correction

Effect of synchronism correction

New synchronism signals are produced by comparing the actual values with the corrected setpoints. Once a correction process has been undertaken, the synchronism signals should be present again.

13.4.6 Delete synchronism correction and NC reset

Variable $AA_COUP_CORR[Sn] returns the value zero for different situations in which the synchronism correct is deleted:

- Once a synchronized spindle coupling has been activated for the following in question with COUPON(..)/COUPONC(..), an existing synchronism correction is adopted in the setpoint position.
- A synchronism correction active during NC reset but not at the parts program end is adopted in the setpoint position. This does not affect the synchronism signals.
- At M30, an existing synchronism correction is retained
- At the user end, the correction value can also be deleted at any early point by describing the variable $AA_COUP_CORR with the **value zero**. The synchronism correction is removed immediately and using a ramp with reduced acceleration rate if larger values are involved.

13.4.7 Special points regarding start-up of a synchronous spindle coupling

Spindle start-up

The leading and following spindles must be started up initially like a normal spindle. The appropriate procedure is described in:
References:
CNC Commissioning Manual: NCK, PLC, Drive
Function Manual Basic Functions; Spindles (S1)

Requirements

The following parameters must then be set for the synchronous spindle pair:

- The machine numbers for the leading and following spindles

  (for permanently configured coupling with channel-specific machine data MD21300 $MC_COUPLE_AXIS_1[n])

- The required coupling mode (setpoint, actual value or speed coupling)

  (for permanently configured coupling with channel-specific machine data MD21310 $MC_COUPLING_MODE_1[n])
- The gear stage(s) of FS and LS for synchronous operation
- The following coupling properties are still applicable for permanently configured synchronous spindle coupling:
  - Block change response in synchronous spindle operation:

    MD21320 $MC_COUPLE_BLOCK_CHANGE_CTRL_1
  - Coupling cancellation response:

    MD21330 $MC_COUPLE_RESET_MODE_1
  - Write-protection for coupling parameters:

    MD21340 $MC_COUPLE_IS_WRITE_PROT_1
  - Transformation parameters for synchronous spindle coupling:

    SD42300 $SC_COUPLE_RATIO_1[n]

Command behavior of FS and LS for setpoint coupling

In order to obtain the best possible synchronism in **setpoint couplings**, the FS and LS must have the same dynamic response as the response to setpoint changes. The axial control loops (position, speed and current controllers) should each be set to the **optimum** value so that variances can be eliminated as quickly and efficiently as possible.

The **dynamic response adaptation** function in the setpoint branch is provided to adapt different axis dynamic responses without loss of control quality (see also Section "K3: Compensations (Page 223)"). The following control parameters must each be set optimally for the FS and LS:

- K_V factor (MD32200 $MA_POSCTRL_GAIN)
- Feedforward control parameters

  MD32620 $MA_FFW_MODE

  MD32610 $MA_VELO_FFW_WEIGHT

  MD32650 $MA_AX_INERTIA

  MD32800 $MA_EQUIV_CURRCTRL_TIME

  MD32810 $MA_EQUIV_SPEEDCTRL_TIME

Behavior during loss of synchronism:

- Axis-specific MD32620 $MA_FFW_MODE

We recommend setting the **feedforward control mode** of the following axis to speed feedforward control with Tt symmetrization MD32620 = 3.

This feedforward control mode can be further optimized for a more secure symmetrization process by changing the axis-specific machine data:

Machine data	Meaning
MD32810 EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control loop for feedforward control
MD37200 COUPLE_POS_TOL_COURSE	Threshold value for "Coarse synchronism"
MD37210 COUPLE_POS_TOL_FINE	Threshold value for "Fine synchronism"
MD37220 COUPLE_VELO_TOL_COURSE	Velocity tolerance 'coarse'
MD37220 COUPLE_VELO_TOL_FINE	Velocity tolerance 'fine'

In such cases, higher threshold values for the synchronism signals and larger position and/or speed tolerances result in more stable results.

Dynamic response adaptation

To obtain a good control behavior, FS and LS must have the same dynamic response. The following error for FS and LS must be equal at any given speed. For dynamically different spindles, a matching via the dynamic response adaptation can be achieved in the setpoint branch. The difference of the equivalent time constants between the dynamic "weakest" spindle to the associated other spindle must be entered as time constant of the dynamic response adaptation.

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.

- Equivalent time constant LS: MD32810 $MA_EQUIV_SPEEDCTRL_TIME[<LS>] = 5 ms
- Equivalent time constant FS: MD32810 $MA_EQUIV_SPEEDCTRL_TIME[<FS>] = 3 ms
- Time constant of dynamic response adaptation for FS: MD32910 $MA_DYN_MATCH_TIME[<FS>] = 5 ms - 3 ms = 2 ms
- Activation of the dynamic response adaptation for FS: MD32900 $MA_DYN_MATCH_ENABLE[<FS>] = 1

The following error for FS and LS is identical for correctly set dynamic response adaptation: Operating area "Diagnostics" > "Service axes"

For the optimization, it may be necessary to adjust servo gain factors or feedforward control parameters slightly.

Position control parameter sets

In the case of spindles, each gear stage is assigned a position-control parameter set. These parameter sets can be used to adapt the dynamic response for LS and FS in synchronous operation. This requires that a different gear stage is used for speed and positioning operation and synchronous operation. The associated gear stage must be switched before activating the associated operating mode.

Control parameters

The following control parameters must be set identically for the FS and LS:

- MD33000 $MA_FIPO_TYPE (fine interpolator type)
- MD32400 $MA_AX_JERK_ENABLE (axial jerk limitation)
- MD32402 $MA_AX_JERK_MODE (filter type for axial jerk limitation)
- MD32410 $MA_AX_JERK_TIME (time constant for the axial jerk filter)
- MD32412 $MA_AX_JERK_FREQ (blocking frequency of the axial jerk filter)
- MD32414 $MA_AX_JERK_DAMP (damping of the axial jerk filter)
- MD32420 $MA_JOG_AND_POS_JERK_ENABLE (release jerk limitation)
- MD32430 $MA_JOG_AND_POS_MAX_JERK (axial jerk)

FS: Automatic parameterization of the control parameters

The control parameters of the following spindle can be set as follows using this machine data:

MD30455 $MA_MISC_FUNCTION_MASK

Bit 5=0: Synchronous spindle coupling, following spindle:

Position control, feedforward control and parameter block are set for the following spindle.

Bit 5=1: Synchronous spindle coupling:

The control parameters of the following spindle are set as in an uncoupled scenario.

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

Automatic transfer of the LS parameters for synchronous operation

For the automatic dynamic response adaptation for FS and LS, for synchronous operation, the parameter values for the position control, feedforward control and parameter records of the FS can be transferred from the LS:

MD30455 $MA_MISC_FUNCTION_MASK, Bit 5

Separate dynamic response for spindle and axis operations

In spindle and axis operations, dynamic programming FA, OVRA, ACC and VELOLIMA can be set separately from one another with the following MD:

MD30455 $MA_MISK_FUNCTION_MASK Bit 6=0

Assignment is undertaken by the programmed axis or spindle name. For example, in spindle operation, VELOLIMA[S1]=50 therefore only reduces the maximum speed to 50% and in axis operation, VELOLIMA[C]=50 only reduces the maximum speed to 50%.

If, for example, VELOLIMA[S1]=50 and VELOLIMA[C]=50 are to have the same effect as before with this machine data, the programming of FA, OVRA, ACC and VELOLIM have an effect regardless of the programmed names:

MD30455 $MA_MISK_FUNCTION_MASK Bit 6=1

Knee-shaped acceleration characteristic

For the leading spindle, the effect of a knee-shaped acceleration characteristic on the following spindle is identified by the following axis-specific machine data:

- MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT (speed for reduced acceleration) and
- MD35230 $MA_ACCEL_REDUCTION_FACTOR (reduced acceleration).

If MD35242 $MA_ACCEL_REDUCTION_TYPE is present, it is also used to configure the type of acceleration reduction. Otherwise a hyperbolic drop in acceleration is assumed.

If the dynamic response of a following spindle is lower than that of the leading spindle when the coupling factor is taken into account, the leading spindle dynamic response is reduced to the required level while the coupling is active.

The acceleration should be **constant** over the entire speed range for the following spindle. However, if a knee-shaped acceleration characteristic is also stored in the above-mentioned machine data for the following spindle, this is only taken into account when the spindles are coupled in. The setpoints of the following spindle are applied for the specified knee-shaped acceleration characteristic.

References:
Function Manual, Basic Functions; Acceleration (B2),
Section: Knee-shaped acceleration characteristic

Actual value coupling

In an actual value coupling (AV), the drive for the FS must be considerably more dynamic than the leading spindle drive. The individual drives in an actual value coupling are also set optimally according to their dynamic response.

An actual value coupling should only be used in exceptional cases.

Speed coupling

The velocity coupling (VV) corresponds internally to a setpoint coupling (DV), but with lower dynamic requirements of the FS and LS. A position control servo loop is not required for the FS and/or LS. Measuring systems are not required.

Threshold values for coarse/fine synchronism

After controller optimization and feedforward control setting, the threshold values for coarse and fine synchronism must be entered for the FS.

- Threshold value for "Coarse synchronism"

  axis-specific MD7200: AV, DV: COUPLE_POS_TOL_COARSE

  MD37220: VV: COUPLE_VELO_TOL_COARSE
- Threshold value for "Fine synchronism"

  axis-specific MD37210: AV, DV: COUPLE_POS_TOL_FINE

  MD37230: VV: COUPLE_VELO_TOL_FINE

The values of the FS must be calculated according to the accuracy requirements of the machine manufacturer, and the PLC interface must be checked via the service display.

Angular offset LS/FS

If there must be a defined angular offset between the FS and LS, e.g. when synchronous operation is activated, the "zero degree positions" of the FS and LS must be mutually adapted. This can be done with the following machine data:

- MD34100 $MA_REFP_SET_POS
- MD34080 $MA_REFP_MOVE_DIST
- MD34090 $MA_REFP_ MOVE_DIST_CORR

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

Service display for FS

In the "Diagnostics" operating area, when commissioning in the synchronous mode, the following values are displayed for the following spindle:

- Actual deviation between setpoints of FS and LS

  Value displayed: Position offset in relation to leading spindle (setpoint)

  (value corresponds to angular offset between FS and LS that can be read with axis variable $AA_COUP_OFFS in the part program)
- Actual deviation between actual values of FS and LS

  Value displayed: Position offset in relation to leading spindle (actual value)

References:
Operating Manual

13.5 Boundary conditions

Availability of the "synchronous spindle" function

The function is an option ("synchronous spindle/multi-edge turning" or the corresponding optional version of the generic coupling), which must be assigned to the hardware via the license management.

Note

Information on the different versions of the generic coupling, refer to:
References:
Function Manual Special Functions; Coupled Axes (M3)

13.6 Examples

Programming example

Program code	Comment
	; Leading spindle = master spindle = spindle 1
	; Following spindle = spindle 2
N05 M3 S3000 M2=4 S2=500	; Master spindle rotates at 3000 rpm
	; FS: 500 rpm
N10 COUPDEF (S2, S1, 1, 1, "No", "Dv")	; Def. of coupling, can also
	; be configured
N70 SPCON	; Include leading spindle in the position control
	; (setpoint value coupling).
N75 SPCON(2)	; Bring following spindle into closed-loop position control
N80 COUPON (S2, S1, 45)	; On-the-fly coupling to offset position = 45 degrees
N200 FA [S2] = 100	; Positioning speed = 100 degrees/min
N205 SPOS[2] = IC(-90)	; Traverse with 90° overlay in negative direction
N210 WAITC(S2, "Fine")	; Wait for "fine" synchronism
N212 G1 X.., Y.. F...	; Machining
N215 SPOS[2] = IC(180)	; Traverse with 180° overlay in positive direction
N220 G4 S50	; Dwell time = 50 revolutions
	; of master spindle

Program code	Comment
N225 FA [S2] = 0	; Activate configured velocity (MD)
N230 SPOS[2] = IC (-7200)	; 20 rev. with configured velocity
	; in neg. direction
N350 COUPOF (S2, S1)	; On-the-fly decoupling, S = S2 = 3000
N355 SPOSA[2] = 0	; Stop FS at zero degrees
N360 G0 X0 Y0	
N365 WAITS(2)	; Wait for spindle 2
N370 M5	; Stop FS
N375 M30	

13.7 Data lists

13.7.1 Machine data

13.7.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB	Machine axis name

13.7.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20070	AXCONF_MACHAX_USED	Machine axis number valid in channel
21300	COUPLE_AXIS_1	Definition of synchronous spindle pair
21310	COUPLING_MODE_1	Type of coupling in synchronous spindle mode
21320	COUPLE_BLOCK_CHANGE_CTRL_1	Block change behavior in synchronous spindle operation
21330	COUPLE_RESET_MODE_1	Coupling abort behavior
21340	COUPLE_IS_WRITE_PROT_1	Coupling parameters are write-protected

13.7.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30455	MISK_FUNCTION_MASK	Axis functions
30550	AXCONF_ASSIGN_MASTER_CHAN	Reset position of channel for axis change
32200	POSCTRL_GAIN	Servo gain factor
32400	AX_JERK_ENABLE	Axial jerk limitation
32410	AX_JERK_TIME	Time constant for axial jerk filter
32420	JOG_AND_POS_JERK_ENABLE	Initial setting for axial jerk limitation
32430	JOG_AND_POS_MAX_JERK	Axial jerk
32610	VELO_FFW_WEIGHT	Feedforward control factor for speed feedforward control
32620	FFW_MODE	Feedforward control 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
34080	REFP_MOVE_DIST	Reference point approach distance
34090	REFP_MOVE_DIST_CORR	Reference point offset
34100	REFP_SET_POS	Reference point value
35000	SPIND_ASSIGN_TO_MACHAX	Assignment of spindle to machine axis
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	Speed tolerance "coarse" between leading and following spindles
37230	COUPLE_VELO_TOL_FINE	Speed tolerance "fine" between leading and following spindles

13.7.2 Setting data

13.7.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42300	COUPLE_RATIO_1	Transmission parameters for synchronous spindle operation

13.7.3 Signals

13.7.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
NC Start	DB21, ...DBX7.1	DB3200.DBX7.1
NC Stop axes plus spindle	DB21, ...DBX7.4	DB3200.DBX7.4

13.7.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Dry run feedrate selected	DB21, ...DBX24.6	DB1700.DBX0.6
Feedrate override selected for rapid traverse	DB21, ...DBX25.3	DB1700.DBX1.3

13.7.3.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Axis/spindle disable	DB31, ...DBX1.3	DB380x.DBX1.3
Follow-up mode	DB31, ...DBX1.4	DB380x.DBX1.4
Position measuring system 1, position measuring system 2	DB31, ...DBX1.5/6	DB380x.DBX1.5/6
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Distance-to-go/Spindle RESET	DB31, ...DBX2.2	DB380x.DBX2.2
Spindle stop/feed stop	DB31, ...DBX4.3	DB380x.DBX4.3
Traversing keys for JOG	DB31, ...DBX4.6/7	DB380x.DBX4.6/7
Re-synchronize spindle 1, re-synchronize spindle 2	DB31, ...DBX16.4/5	DB380x.DBX2000.4/5
Delete S value	DB31, ...DBX16.7	DB380x.DBX2000.7
Feedrate override valid	DB31, ...DBX17.0	DB380x.DBX2001.0
Invert M3/M4	DB31, ...DBX17.6	DB380x.DBX2001.6
Spindle override	DB31, ...DBB19	DB380x.DBB2003
Re-synchronizing	DB31, ...DBX31.4	-
Disable synchronization	DB31, ...DBX31.5	-

13.7.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized 1, referenced/synchronized 2	DB31, ...DBX60.4/5	DB390x.DBX0.4/5
Synchronous mode	DB31, ...DBX84.4	DB390x.DBX2002.4
Synchronism fine	DB31, ...DBX98.0	-
Synchronism coarse	DB31, ...DBX98.1	-
Actual value coupling	DB31, ...DBX98.2	-
Superimposed motion	DB31, ...DBX98.4	DB390x.DBX5002.4
Leading spindle LS/LA active	DB31, ...DBX99.0	-
Following spindle FS/FA active	DB31, ...DBX99.1	-

13.7.4 System variables

System variable	Description
$P_COUP_OFFS[following spindle]	Programmed offset of the synchronous spindle
$AA_COUP_OFFS[following spindle]	Position offset for synchronous spindle (setpoint)
$VA_COUP_OFFS[following spindle]	Position offset for synchronous spindle (actual value)

For a more detailed description of system variables, see:
References:
List Manual System Variables

S7: Memory configuration 14

14.1 Brief description

Memory types

To store and manage data, the NC requires a static memory and a dynamic memory:

- Static NC memory

  In the static NC memory, the program data (part programs, cycles, etc.) and the current system and user data (tool management, global user data, etc.) is saved to persistent memory.

- Dynamic NC memory

  In the dynamic NC memory the data is saved to non-persistent memory. The data here concerns information generated by the NC that is only required for a limited time (e.g., macros, local user data, interpolation buffer, etc.).

Memory organization

The memory areas of the individual data groups in the static and dynamic NC memories have defined sizes, which are fixed when the memory is configured.

This type of memory organization ensures the deterministic behavior of the control: The reserved memory area is guaranteed throughout part program processing.

Memory configuration

When booted for the first time, the system enters default values in all other memory-configuration machine data. This configuration is adequate in most cases. However, the machine manufacturer/user can make changes at any time (Reconfiguration).

14.2 Memory organization

14.2.1 Active and passive file system

The static NC memory contains an active and passive file system.

Active file system

The active file system contains system data used to parameterize the NCK:

- Machine data
- Setting data
- Option data
- Global user data (GUD)
- Tool-offset/magazine data
- Protection zones
- R parameters
- Work offsets/FRAME
- Sag compensations
- Quadrant error compensation
- Leadscrew error compensation

This data represents the current work data of the NCK.

The user's view of the active file system is data-oriented.

Passive file system

The passive file system contains all files loaded onto the NCK:

- Main programs
- Subprograms
- Global-user-data and macro definition files (*.DEF)
- Standard cycles
- User cycles
- Workpieces
- Comments

The user's view of the passive file system is file-oriented.

14.2.2 Reconfiguration

Reconfiguration

NOTICE
Data loss
A reconfiguration of the **static** NC memory results in a loss of data on the active and passive file system. Before activating the modified memory configuration, you must must first save the data by creating a **series machine start-up file**.

The following actions result in the reconfiguration of the static and/or dynamic NC memory:

- Changing the settings of memory-configuration machine data:

  MD... $..._MM_...
- Changing the number of channels

14.3 Configuration of the static user memory

14.3.1 Division of the static NC memory

The figure below shows the division of the static NC memory for SINUMERIK 840D sl:

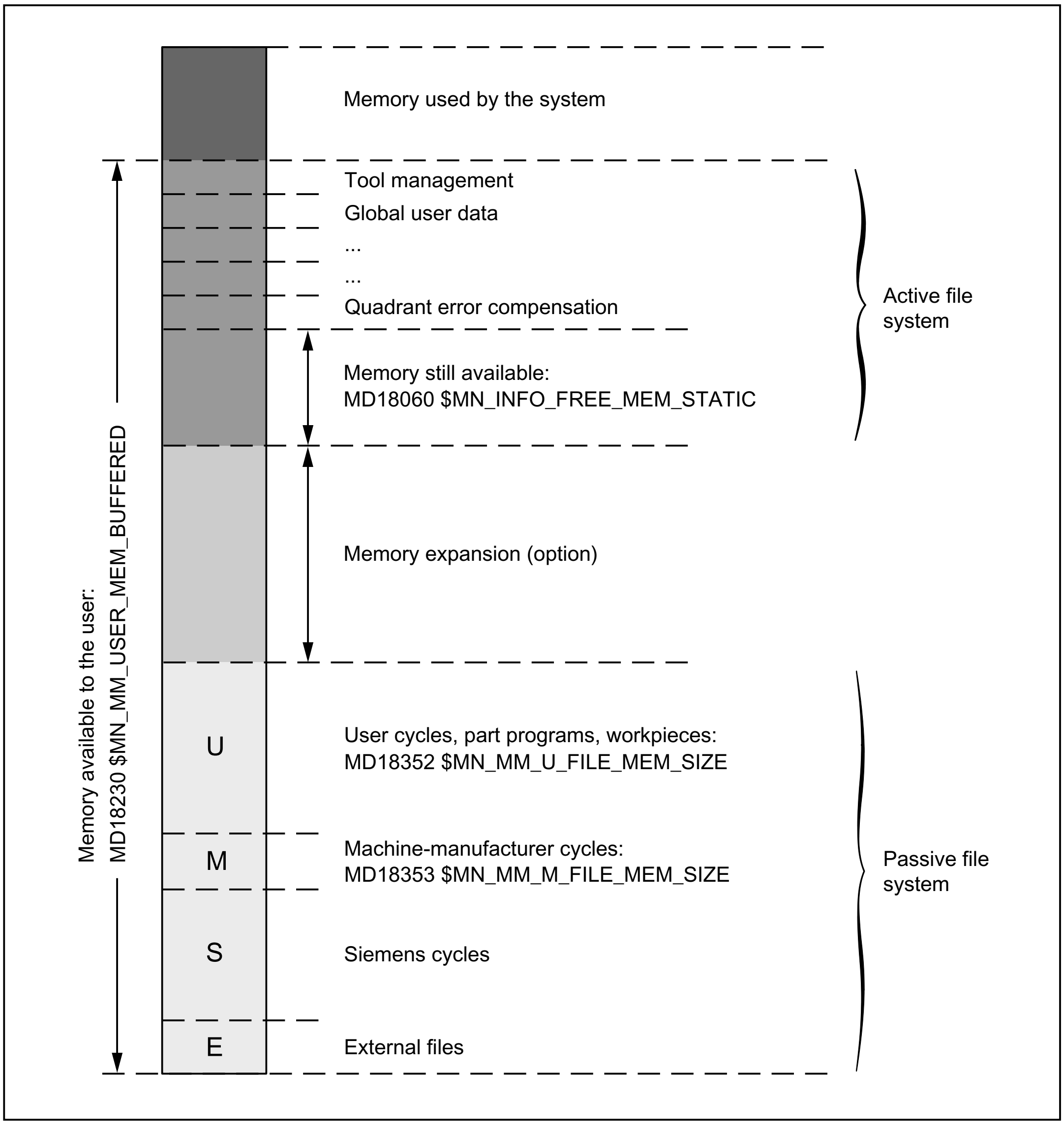


Figure 14-1 Static NC memory for SINUMERIK 840D sl

Static user memory

The static NC memory is used jointly by the system and by the user.

The area available to the user is defined as the static user memory. It contains the data from the active and passive file system.

Static-user-memory size

The size of the static user memory is defined in machine data:

MD18230 $MN_MM_USER_MEM_BUFFERED

Memory space for passive file system

The memory space for the passive file system has a defined size and is divided into the following partitions:

Partition	Storage of:
S (**S**iemens = Control manufacturer)	Files from the _N_CST_DIR directory (Siemens cycles)
M (**M**anufacturer = Machine manufacturer)	Files from the _N_CMA_DIR directory (Machine-manufacturer cycles)
U (**U**ser = End customer)	Files from the _N_CUS_DIR directory (User cycles, part programs, workpieces)
E (**E**XT = External)	External files (e.g., part programs from USB FlashDrive or CD). **Note:** To avoid naming conflicts with internal part programs, part programs installed from external data carriers are stored in a separate directory (_N_EXT_DIR).

Advantage of separation:

This separation into different partitions ensures that a series machine start-up file can be reloaded onto the NCK even in the event of an NCK software or cycle package upgrade (that has expanded the area of Siemens cycles).

Size of the partitions:

The size of partitions **S** and **E** are preset and cannot be modified.

You can divide the remaining memory available for the passive file system into the partitions **U** and **M** as you see fit. The settings are made with machine data:

MD18352 $MN_MM_U_FILE_MEM_SIZE
(end-user memory for part programs/cycles/files)

MD18353 $MN_MM_M_FILE_MEM_SIZE
(memory size for cycles/files of the machine manufacturer)

The maximum adjustable values depend on:

- The system and thus the memory space available (including an optional memory expansion)
- from the defined maximum values, see also:

  References:
  Detailed machine-data description

Memory space for active file system

The memory space for the active file system is divided into various data areas (tool management, global user data, etc.), which can be defined individually using machine data.

The memory still available is shown in machine data:

MD18060 $MN_INFO_FREE_MEM_STATIC
(free-static-memory display data)

You can expand the sizes of the individual memory areas for the active file system with the relevant machine data until the available memory has been used.

Note

The memory required to expand the memory areas is displayed in the "Startup" area of the user interface. This information enables the system startup engineer to estimate the actual memory requirements for the planned expansion.

Memory expansion (option)

The machine manufacturer can also purchase additional static user memory as an option.

You can use the additional memory space as required to expand partitions U and M or to expand the memory area of the active file system.

14.3.2 Commissioning

Procedure

1. Load standard machine data.
2. Preset machine data:
   MD18230 $MN_MM_USER_MEM_BUFFERED
   with a high value (> default memory available + optional additional memory).
3. Reset the NCK.

   Alarm 6030 "User memory limit adjusted" is triggered and the maximum memory available for the user is entered in MD18230 (including optional memory expansion). Default values are entered in all other memory-configuration machine data.
4. Set the sizes of partitions U and M in machine data:

   MD18352 $MN_MM_U_FILE_MEM_SIZE
   (end-user memory for part programs/cycles/files)

   MD18353 $MN_MM_M_FILE_MEM_SIZE
   (memory size for cycles/files of the machine manufacturer)
5. Activate the number of required channels and axes.
6. You can adjust the default memory division by increasing/decreasing individual memory areas of the active file system (tool management, global user data, etc.) for each user.
   - The static user memory still available is displayed in machine data:

     MD18060 $MN_INFO_FREE_MEM_STATIC
     (free-static-memory display data)
   - Set machine data (see Section "Data lists (Page 764)").

     References:
     Detailed machine-data description
7. Reset the NCK.

   The memory is set up again.

14.4 Configuration of the dynamic user memory

14.4.1 Division of the dynamic NC memory

The figure below shows the division of the dynamic NC memory:

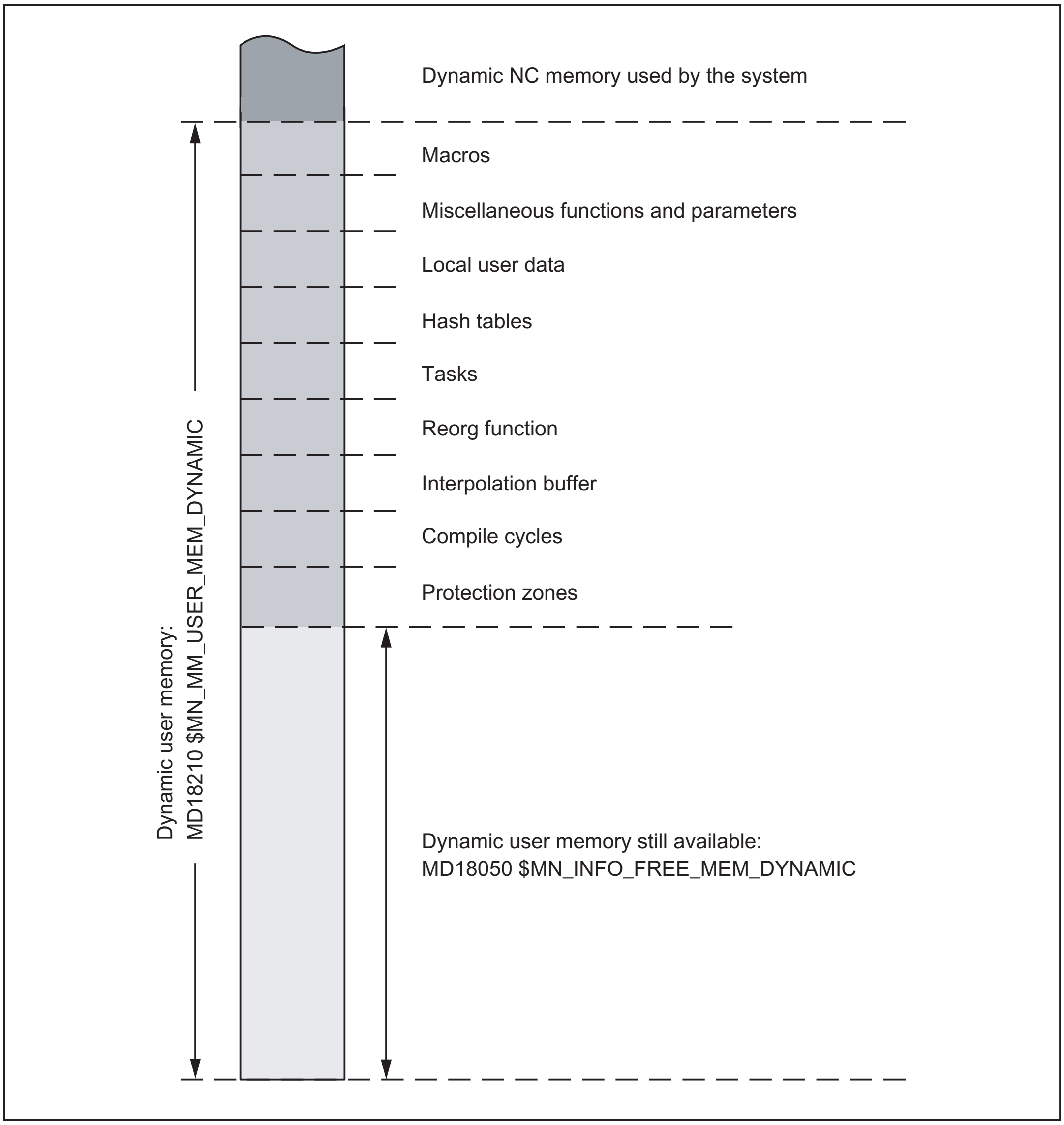


Figure 14-2 Dynamic NC memory

Dynamic user memory

The dynamic NC memory is used jointly by the system and by the user. The area available to the user is defined as the dynamic user memory.

Dynamic-user-memory size

The size of the dynamic user memory is set in machine data:

MD18210 $MN_MM_USER_MEM_DYNAMIC

Changes are not usually required as an appropriate value is automatically set during the reconfiguration.

Dynamic user memory still available

The dynamic memory still available is shown in machine data:

MD18050 $MN_INFO_FREE_MEM_DYNAMIC
(free-dynamic-memory display data)

The content of this machine data indicates how much memory is available to expand the user data areas (local user data, IPO buffer, etc.) for each channel.

14.4.2 Commissioning

You can adjust the default memory division by increasing/decreasing individual memory areas for each user.

Procedure

1. The dynamic user memory still available is displayed in machine data:

   MD18050 $MN_INFO_FREE_MEM_DYNAMIC
   (free-dynamic-memory display data)

2. Set machine data (see Section "Data lists (Page 764)").

   References:
   Detailed machine-data description

3. Reset the NCK.

   The memory is set up again.

14.5 Data lists

14.5.1 Machine data

14.5.1.1 General machine data

Number	Identifier: $MN_	Description
10134	MM_NUM_MMC_UNITS	Number of simultaneous HMI communication partners
10850	MM_EXTERN_MAXNUM_OEM_GCODES	Maximum number of OEM-G codes
10880	MM_EXTERN_CNC_SYSTEM	Definition of the control system to be adapted
10881	MM_EXTERN_GCODE_SYSTEM	ISO_3 Mode: GCodeSystem
18050	INFO_FREE_MEM_DYNAMIC	Free-dynamic-memory display data
18060	INFO_FREE_MEM_STATIC	Free-static-memory display data
18070	INFO_FREE_MEM_DPR	Display data for free memory in dual-port RAM
18072	INFO_FREE_MEM_CC_MD	Displays memory available in the CC-MD memory
18078	MM_MAX_NUM_OF_HIERARCHIES	Maximum number of definable magazine-location-type hierarchies
18079	MM_MAX_HIERARCHY_ENTRIES	Maximum permissible number of entries in a magazine-location-type hierarchy
18080	MM_TOOL_MANAGEMENT_MASK	Mask for reserving memory for tool management
18082	MM_NUM_TOOL	Number of tools managed by NCK
18084	MM_NUM_MAGAZINE	Number of magazines managed by NCK
18086	MM_NUM_MAGAZINE_LOCATION	Number of magazine locations
18088	MM_NUM_TOOL_CARRIER	Maximum number of definable toolholders
18090	MM_NUM_CC_MAGAZINE_PARAM	Compile-cycle tool management: quantity of magazine data
18092	MM_NUM_CC_MAGLOC_PARAM	Compile-cycle tool management: quantity of magazine-location data
18094	MM_NUM_CC_TDA_PARAM	Compile-cycle tool management: quantity of TDA data
18096	MM_NUM_CC_TOA_PARAM	Compile-cycle tool management: quantity of TOA data
18098	MM_NUM_CC_MON_PARAM	Compile-cycle tool management: quantity of monitor data
18100	MM_NUM_CUTTING_EDGES_IN_TOA	Number of tool offsets in NCK
18102	MM_TYPE_OF_CUTTING_EDGE	Type of D-number programming
18104	MM_NUM_TOOL_ADAPTER	Tool adapter in TO area
18105	MM_MAX_CUTTING_EDGE_NO	Maximum value of D number
18106	MM_MAX_CUTTING_EDGE_PERTOOL	Maximum number of D numbers per tool
18108	MM_NUM_SUMCORR	Additive offsets in the TO area

Number	Identifier: $MN_	Description
18110	MM_MAX_SUMCORR_PER_CUTTEDGE	Maximum number of sum offsets per cutting edge
18112	MM_KIND_OF_SUMCORR	Properties of additive offsets in the TO area
18114	MM_ENABLE_TOOL_ORIENT	Assign orientation to cutting edges
18116	MM_NUM_TOOL_ENV	Number of tool environments in TO area
18118	MM_NUM_GUD_MODULES	Number of GUD modules
18120	MM_NUM_GUD_NAMES_NCK	Number of global user variables
18130	MM_NUM_GUD_NAMES_CHAN	Number of channel-specific user variables
18140	MM_NUM_GUD_NAMES_AXIS	Number of axis-specific user variables
18150	MM_GUD_VALUES_MEM	Memory space for global user variables
18160	MM_NUM_USER_MACROS	Number of macros
18170	MM_NUM_MAX_FUNC_NAMES	Number of miscellaneous functions
18180	MM_NUM_MAX_FUNC_PARAM	Number of additional parameters
18190	MM_NUM_PROTECT_AREA_NCK	Number of protection zones in NCK
18200	MM_NUM_CCS_MAGAZINE_PARAM	Number of Siemens OEM magazine data
18201	MM_TYPE_CCS_MAGAZINE_PARAM	Siemens OEM magazine data type
18202	MM_NUM_CCS_MAGLOC_PARAM	Number of Siemens OEM magazine location data
18203	MM_TYPE_CCS_MAGLOC_PARAM	Siemens OEM magazine location data type
18204	MM_NUM_CCS_TDA_PARAM	Number of Siemens OEM tool data
18205	MM_TYPE_CCS_TDA_PARAM	Siemens OEM tool data type
18206	MM_NUM_CCS_TOA_PARAM	Number of Siemens OEM data per cutting edge
18207	MM_TYPE_CCS_TOA_PARAM	Siemens OEM data type per cutting edge
18208	MM_NUM_CCS_MON_PARAM	Number of Siemens OEM monitor data
18209	MM_TYPE_CCS_MON_PARAM	Siemens OEM monitor data type
18210	MM_USER_MEM_DYNAMIC	User memory in DRAM
18220	MM_USER_MEM_DPR	User memory in dual-port RAM
18230	MM_USER_MEM_BUFFERED	User memory in SRAM
18231	MM_USER_MEM_BUFFERED_TYPEOF	Data-buffer technology
18232	MM_ACTFILESYS_LOG_FILE_MEM	System: Log-file size
18238	MM_CC_MD_MEM_SIZE	Machine-data compile cycles in SRAM
18240	MM_LUD_HASH_TABLE_SIZE	Hash-table size for user variables
18242	MM_MAX_SIZE_OF_LUD_VALUE	Maximum LUD-variable array size
18250	MM_CHAN_HASH_TABLE_SIZE	Hash-table size for channel-specific data
18260	MM_NCK_HASH_TABLE_SIZE	Hash-table size for global data
18270	MM_NUM_SUBDIR_PER_DIR	Number of subdirectories
18280	MM_NUM_FILES_PER_DIR	Number of files per directory
18290	MM_FILE_HASH_TABLE_SIZE	Hash-table size for files in a directory
18300	MM_DIR_HASH_TABLE_SIZE	Hash-table size for subdirectories
18310	MM_NUM_DIR_IN_FILESYSTEM	Number of directories in passive file system

Number	Identifier: $MN_	Description
18320	MM_NUM_FILES_IN_FILESYSTEM	Number of files in passive file system
18332	MM_FLASH_FILE_SYSTEM_SIZE	Size of flash file system on PCNC
18342	MM_CEC_MAX_POINTS	Maximum table size for sag compensation
18350	MM_USER_FILE_MEM_MINIMUM	Minimum part-program memory
18352	MM_U_FILE_MEM_SIZE	End-user memory for part programs/cycles/files
18353	MM_M_FILE_MEM_SIZE	Memory size for cycles/files of the machine manufacturer
18354	MM_S_FILE_MEM_SIZE	Memory size for cycles/files of the NC manufacturer
18355	MM_T_FILE_MEM_SIZE	Memory size for temporary files
18356	MM_E_FILE_MEM_SIZE	Memory size for external files
18360	MM_EXT_PROG_BUFFER_SIZE	FIFO buffer size for execution from external source (DRAM)
18362	MM_EXT_PROG_NUM	Number of program levels that can be processed simultaneously from external
18370	MM_PROTOC_NUM_FILES	Maximum number of log files
18371	MM_PROTOC_NUM_ETPD_STD_LIST	Number of standard ETPD data lists
18372	MM_PROTOC_NUM_ETPD_OEM_LIST	Number of ETPD OEM data lists
18373	MM_PROTOC_NUM_SERVO_DATA	Number of servo data for log
18374	MM_PROTOC_FILE_BUFFER_SIZE	Log-file buffer size
18375	MM_PROTOC_SESS_ENAB_USER	User-enabled for sessions
18390	MM_COM_COMPRESS_METHOD	Supported compression method
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 (DRAM)
18408	MM_NUM_CURVE_SEGMENTS_DRAM	Number of curve segments (DRAM)
18409	MM_NUM_CURVE_SEG_LIN_DRAM	Number of linear curve segments (DRAM)
18410	MM_NUM_CURVE_POLYNOMS_DRAM	Number of curve table polynomials (DRAM)
18450	MM_NUM_CP_MODULES	Maximum number of CP modules
18452	MM_NUM_CP_MODUL_LEAD	Maximum number of leading values per CP coupling module
18500	MM_EXTCOM_TASK_STACK_SIZE	Stack size of external communication task (DRAM)
18502	MM_COM_TASK_STACK_SIZE	Stack size in Kbytes of communication task (DRAM)
18510	MM_SERVO_TASK_STACK_SIZE	Stack size of servo task (DRAM)
18512	MM_IPO_TASK_STACK_SIZE	Stack size of IPO task (DRAM)
18520	MM_DRIVE_TASK_STACK_SIZE	Stack size of drive task (DRAM)
18540	MM_PLC_TASK_STACK_SIZE	Stack size of PLC task (DRAM)
18600	MM_FRAME_FINE_TRANS	Fine offset for FRAME (SRAM)
18601	MM_NUM_GLOBAL_USER_FRAMES	Number of globally predefined user frames (SRAM)

Number	Identifier: $MN_	Description
18602	MM_NUM_GLOBAL_BASE_FRAMES	Number of global basic frames (SRAM)
18660	MM_NUM_SYNACT_GUD_REAL	Number of configurable real-type GUD variables
18661	MM_NUM_SYNACT_GUD_INT	Number of configurable integer-type GUD variables
18662	MM_NUM_SYNACT_GUD_BOOL	Number of configurable Boolean-type GUD variables
18663	MM_NUM_SYNACT_GUD_AXIS	Number of configurable axis-type GUD variables
18664	MM_NUM_SYNACT_GUD_CHAR	Configurable char-type GUD variable
18665	MM_NUM_SYNACT_GUD_STRING	Configurable STRING-type GUD variable
18700	MM_SIZEOF_LINKVAR_DATA	Size of the NCU link variable memory
18710	MM_NUM_AN_TIMER	Number of global time variables for synchronized actions
18720	MM_SERVO_FIFO_SIZE	Setpoint for buffer size between IPO and closed-loop position control
18780	MM_NCU_LINK_MASK	Activation of NCU link communication
18781	NCU_LINK_CONNECTIONS	Number of internal link connections
18782	MM_LINK_NUM_OF_MODULES	Number of NCU link modules
18790	MM_MAX_TRACE_LINK_POINTS	Size of trace data buffer for NCU link
18792	MM_TRACE_LINK_DATA_FUNCTION	Specifies the contents of NCU link files
18794	MM_TRACE_VDI_SIGNAL	Trace specification of VDI signals
18800	MM_EXTERN_LANGUAGE	Activation of external NC languages
18860	MM_MAINTENANCE_MON	Activate recording of maintenance data
18870	MM_MAXNUM_KIN_CHAINS	Maximum number of trains
18880	MM_MAXNUM_KIN_CHAIN_ELEM	Maximum number of train elements
18890	MM_MAXNUM_3D_PROT_AREAS	Maximum number of elements in 3D protection zones
18892	MM_MAXNUM_3D_PROT_AREA_ELEM	Maximum number of protection-zone elements
18894	MM_MAXNUM_3D_PROT_GROUPS	Maximum number of protection-zone groups
18896	MM_MAXNUM_3D_COLLISION	Maximum number of temporary memory locations for collision check

14.5.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20096	T_M_ADDRESS_EXIT_SPINO	Spindle number as address extension
27900	REORG_LOG_LIMIT	Percentage of IPO buffer for log-file enable
28000	MM_REORG_LOG_FILE_MEM	Memory size for REORG
28010	MM_NUM_REORG_LUD_MODULES	Number of modules for local user variables with REORG
28020	MM_NUM_LUD_NAMES_TOTAL	Number of local user variables
28040	MM_LUD_VALUES_MEM	Memory size for local user variables
28050	MM_NUM_R_PARAM	Number of channel-specific R parameters

Number	Identifier: $MC_	Description
28060	MM_IPO_BUFFER_SIZE	Number of NC blocks in the IPO buffer
28070	MM_NUM_BLOCKS_IN_PREP	Number of blocks for block preparation
28080	MM_NUM_USER_FRAMES	Number of settable frames
28081	MM_NUM_BASE_FRAMES	Number of basic frames (SRAM)
28082	MM_SYSTEM_FRAME_MASK	System frames (SRAM)
28083	MM_SYSTEM_DATAFRAME_MASK	System frames (SRAM)
28085	MM_LINK_TOA_UNIT	Allocation of a TO unit to a channel
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of block elements for compile cycles
28100	MM_NUM_CC_BLOCK_USER_MEM	Size of block memory for compile cycles
28105	MM_NUM_CC_HEAP_MEM	Heap memory in KB for compile cycle applications (DRAM)
28150	MM_NUM_VDIVAR_ELEMENTS	Number of elements for writing PLC variables
28160	MM_NUM_LINKVAR_ELEMENTS	Number of write elements for the NCU link variables
28180	MM_MAX_TRACE_DATAPOINTS	Size of trace data buffer
28200	MM_NUM_PROTECT_AREA_CHAN	Number of modules for channel-specific protection zones
28210	MM_NUM_PROTECT_AREA_ACTIVE	Number of simultaneously active protection zones
28212	MM_NUM_PROTECT_AREA_CONTOUR	Elements for active protection zones (DRAM)
28250	MM_NUM_SYNC_ELEMENTS	Number of elements for expressions in synchronized actions
28252	MM_NUM_FCTDEF_ELEMENTS	Number of FCTDEF elements
28254	MM_NUM_AC_PARAM	Dimension of $AC_PARAM.
28255	MM_BUFFERED_AC_PARAM	$AC_PARAM[ ] is saved in SRAM.
28256	MM_NUM_AC_MARKER	Dimension of $AC_MARKER
28257	MM_BUFFERED_AC_MARKER	$AC_MARKER[ ] is saved in SRAM.
28258	MM_NUM_AC_TIMER	Number of $AC_TIMER time variables (DRAM)
28274	MM_NUM_AC_SYSTEM_PARAM	Number of $AC_SYSTEM_ PARAM for motion-synchronous actions
28276	MM_NUM_AC_SYSTEM_MARKER	Number of $AC_SYSTEM_ MARKER for motion-synchronous actions
28290	MM_SHAPED_TOOLS_ENABLE	Enable tool-radius compensation for contour tools
28300	MM_PROTOC_USER_ACTIVE	Activate logging for a user
28301	MM_PROTOC_NUM_ETP_OEM_TYP	Number of ETP OEM event types
28302	MM_PROTOC_NUM_ETP_STD_TYP	Number of ETP standard event types
28400	MM_ABSBLOCK	Activate block display with absolute values
28402	MM_ABSBLOCK_BUFFER_CONF	Dimension size of upload buffer
28450	MM_TOOL_DATA_CHG_BUFF_SIZE	Buffer for tool data changes (DRAM)
28500	MM_PREP_TASK_STACK_SIZE	Stack size of preparatory task
28520	MM_MAX_AXISPOLY_PER_BLOCK	Maximum number of axis polynomials per block
28530	MM_PATH_VELO_SEGMENTS	Number of memory chips for limiting the tool-path velocity

Number	Identifier: $MC_	Description
28535	MM_FEED_PROFILE_SEGMENTS	Number of memory chips for feed profiles
28540	MM_ARCLENGTH_SEGMENTS	Number of memory chips for displaying the arc length function
28560	MM_SEARCH_RUN_RESTORE_MODE	Restore data after a simulation
28580	MM_ORIPATH_CONFIG	Setting for ORIPATH tool orientation trajectory referred to path

14.5.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
38000	MM_ENC_COMP_MAX_POINTS	Number of intermediate points with interpolatory compensation
38010	MM_QEC_MAX_POINTS	Number of values for quadrant-error compensation

T1: Indexing axes 15

15.1 Brief Description

Indexing axes in machine tools

In certain applications, the axis is only required to approach specific grid points (e.g. location numbers). It is necessary to approach the defined grid points, the indexing positions, both in AUTOMATIC and set-up mode.

The relevant axes are called "indexing axes". The positions defined on the indexing axes are known as "coded positions" or "indexing positions".

Applications

Indexing axes are used predominantly in connection with specific types of tool magazines such as tool turrets, tool chain magazines or tool cartridge magazines. The coded positions refer to the individual locations of the tools in the magazines. During a tool change, the magazine is positioned at the location containing the tool to be loaded.

Display indexing

The following data can be queried via system variables:

- The number of the current indexing position:
  - When the "exact stop fine" window of the indexing position is reached
  - When half the distance to the next indexing position is crossed
- The programmed indexing position

15.2 Traversing of indexing axes

Indexing axes can be traversed:

- Manually in the setting-up modes JOG and INC
- from one part program with special instructions for coded positions
- of PLC

Upon reaching the indexing position the following interface signal is given out to the PLC:

DB31, ... DBX76.6 (indexing axis in position)

15.2.1 Traversing of indexing axes in the JOG mode

Reference point approach

An indexing axis approaches the reference point in the same way as other axes. The reference point does not have to coincide with an indexing position.

When reference point is reached:
DB31, ... DBX60.4 or 5 (referenced/synchronized 1 or 2) = 1
the indexing axis moves only to indexing positions in JOG mode during conventional and incremental traversing.

Exception: No indexing positions are approached when traversing with the handwheel.

If the axis is not referenced (DB31, ... DBX60.4 or 5 = 0), the indexing positions are ignored when traversing in JOG mode!

Note

Hirth indexing axes cannot be traversed in JOG mode before reference point approach.

Continuous traversal in JOG

- Jog mode active:

  SD41050 $SN_JOG_CONT_MODE_LEVELTRIGGRD = 1

  Pressing a "+" or "-" traversing key causes the indexing axis to move in the same way as with conventional JOG traversing. When the traversing key is released, the indexing axis traverses to the next indexing position in the direction of traversing.
- Continuous mode active:

  SD41040 $SN_JOG_CONT_MODE_LEVELTRIGGRD = 0

  Pressing the traversing key briefly (first rising signal edge) starts the traversing movement of the indexing axis in the desired direction. Traversing continues when the traversing key is released. When the traversing key is pressed again (second rising signal edge), the indexing axis traverses to the next indexing position in the direction of traversing.

Indexing axes are generally traversed in JOG mode (standard setting). Continuous mode plays a less important role.

If the operator changes the direction of traversing before the indexing position has been reached, the indexing axis is positioned on the next indexing position in the direction of traversing. The traversing movement must be started in the opposite direction.

For further information on continuous traversing in JOG or continuous mode (see Section "H1: Manual and handwheel travel (Page 143)").

Incremental traversal in JOG mode (INC)

Irrespective of the current increment setting (INC1, ... ,INCvar), the indexing axis always traverses **through one indexing position** in the selected direction when a traversing key "+" or "-" is pressed .

In jog mode, the traversing movement is interrupted when the traversing key is released. The indexing position can be approached by pressing the traversing key again.

In continuous mode, the traversing movement is interrupted when the traversing key is pressed again. The indexing axis is, in this case, not located on the indexing position.

Between indexing positions

If an indexing axis is situated between 2 indexing positions, then it approaches the next-higher indexing position when the "+" traversing key is pressed in JOG-INC mode. Similarly, pressing the "-" traversing key causes the next-lower indexing position to be approached.

Handwheel traversal

When the indexing axis is traversed by means of the handwheel in JOG mode, the **indexing positions are ignored**. As the handwheel is turned, the indexing axis moves to any position depending on the basic system in mm, inches or degrees.

If traversing of the indexing axis with the handwheel is to be interlocked, this can be handled by the PLC user program.

Signal from PLC "Indexing axis in position"

During the traversing motion of the indexing axis in the JOG mode, the following NC/PLC interface signal displays the reaching of the indexing position:

DB31, ... DBX76.6 (indexing axis in position)

Requirement: The indexing axis is referenced (DB31, ... DBX60.4 or 5 = 1)

Alarms in JOG mode

If the indexing axis leaves the traversing range defined in the indexing position table when traversing in JOG mode, alarm 20054 "wrong index for indexing axis in JOG" is output.

Revolutional feedrate

In JOG mode, the response of the axis / spindle also depends on the setting data:

SD41100 $SN_JOG_REV_IS_ACTIVE (revolutional feed rate for JOG active)

SD41100	Meaning
= 1 (active)	The axis / spindle is always traversed with revolutional feed rate as a function of the master spindle: MD32050 $MA_JOG_REV_VELO (revolutional feed rate for JOG mode) or MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feed rate for JOG with rapid traverse override)

SD41100	Meaning
= 0 (not active)	The response of the axis / spindle depends on the setting data: SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE(revolutional feed feed rate for position axes / spindles) The response of a geometry axis on which a frame acts is to rotate, depending on the setting data: SD42600 $SC_JOG_FEED_PER_REV_SOURCE

15.2.2 Traversing of indexing axes in the AUTOMATIC mode

Traversal to selected positions

An axis defined as an indexing axis can be made to approach **any selected position** from the NC part program in AUTOMATIC mode. This includes positions between the defined indexing positions.

These positions are programmed, in the usual way, in the unit of measurement (mm/inches or degrees) for the axis. The general programming instructions used for this purpose (`G90`, `G91`, `AC`, `IC`, etc.) are described in the Programming Manuals.

Traversal to "Coded positions"

Special instructions can also be used in the part program to traverse indexing axes to the "coded positions":

Statement	Effect
`CAC`	Approach absolute coded position
`CACP`	Approach absolute coded position in positive direction
`CACN`	Approach absolute coded position in negative direction
`CIC`	Approach incremental coded position
`CDC`	Approach coded position along direct (shortest) path

With absolute positioning, the indexing position to be approached is programmed, and with incremental positioning, the number of indexes to be traversed in the "+" or "-" direction is programmed.

On rotary axes, the indexing position can be approached directly across the shortest path (`CDC`) or with a defined direction of rotation (`CACP`, `CACN`).

Reaching the indexing position

If the "Exact stop fine" window is reached and the indexing axis is positioned on an indexing position, the following NC/PLC interface signal is enabled regardless of how the indexing position was reached.

DB31, ... DBX76.6 (indexing axis in position)

15.2.3 Traversing of indexing axes by PLC

Indexing axes can also be traversed from the PLC user program.

There are various methods:

- Concurrent positioning axes

  The indexing position to be approached can be specified by the PLC (see Section "P2: Positioning axes (Page 599)").

- Asynchronous subprograms (ASUBs)

  References:
  Function Manual, Basic Functions; Mode Group, Channel, Program Operation, Reset Response (K1)

15.3 Parameterization of indexing axes

Definition of the indexing axis

An axis (linear or rotary axis) can be defined as indexing axis with the axial machine data:

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB

Value	Meaning
0	The axis is not declared as an indexing axis.
1	The axis is an indexing axis. The associated indexing positions are stored in the indexing positions table 1.
2	The axis is an indexing axis. The associated indexing positions are stored in the indexing positions table 2.

Indexing position tables

The axis positions (in mm or degrees) assigned to the indexes must be stored for each indexing axis in the form of a table in machine data.

2 indexing position tables are possible:

MD10910 $MN_INDEX_AX_POS_TAB_1 [n] (indexing position table 1)

MD10930 $MN_INDEX_AX_POS_TAB_2 [n] (indexing position table 2)

Note

Several axes can be assigned to an indexing position table. On condition that these indexing axes are of the same type (linear axis, rotary axis, modulo 360° function).

No. of indexing positions

Up to 60 positions can be entered in each indexing position table:

[n = 0 ... 59]

The actually used number of entries is defined with the machine data:

MD10900 $MN_INDEX_AX_LENGTH_POS_TAB_1 ((number of positions of indexing position table 1)

MD10920 $MN_INDEX_AX_LENGTH_POS_TAB_2 ((number of positions of indexing position table 2)

Note

Entries in the indexing positions table that cross the parameterized number of indexing positions (MD10900 or MD10920) are not considered.

Valid measuring system

The indexing positions defined with MD10900 and MD10920 are related to the measuring system configured for position tables:

MD10270 $MN_POS_TAB_SCALING_SYSTEM

Value	System of units
0	Metric
1	inch

Note

MD10270 has an effect on the following setting data:

SD41500 $SN_SW_CAM_MINUS_POS_TAB_1 (switching point for falling cam edge 1-8)

...

SD41507 $SN_SW_CAM_PLUS_POS_TAB_4 (switching point for rising cam edge 25-32)

Entry for indexing positions

The following rules apply:

- The indexing positions must be entered in the table in ascending order (starting with the negative to the positive traversing range) with no gaps between the entries.
- Consecutive position values cannot be identical.
- The axis positions must be entered in the basic coordinate system.

Modulo rotary axis as indexing axis

The indexing axis is defined with Modulo 360° as rotary axis:

MD30300 $MA_IS_ROT_AX = 1

and

MD30310 $MA_ROT_IS_MODULO = 1

In this case, the following points must be observed additionally for the specification of the indexing positions:

- Permissible range: 0° ≤ Pos < 360°
- Since the indexing axis is defined as a continuously rotating rotary axis, indexing position 1 is approached after the highest valid indexing position in the table has been reached and the axis continues to traverse in the positive direction with INC. Similarly, indexing position 1 is followed by the highest valid indexing position in the negative direction with INC.

15.4 Programming of indexing axes

Coded position

To allow indexing axes to be positioned from the NC part program, special instructions are provided with which the indexing numbers (e.g. location numbers) are programmed instead of axis positions in mm or degrees. The availability of a special instruction depends on the axis type (linear or rotary axis):

Statement	Effect	Availability
`CAC(i)`	Traverse coded position in absolute terms	Linear axis, rotary axis
`CACP(i)`	Traverse coded position ain absolute terms in the positive direction	Rotary axis
`CACN(i)`	Traverse coded position ain absolute terms in the negative direction	Rotary axis
`CDC(i)`	Traverse coded position along the direct (shortest) path	Rotary axis
`CIC(i)`	Traverse coded position incrementally	Linear axis, rotary axis

i:	Coded position (indexing position)
Value range of i:	0 ... 59; whole number (positive and negative values are possible in `CIC`)

Examples

Program code	Comment
POS[B]=CAC(20)	; Indexing axis B approaches the coded position (indexing) 20 in **absolute** mode. The direction of traversing depends on the current actual position.

Program code	Comment
POS[B]=CACP(10)	; Indexing axis B approaches the coded position (index position) 10 in **absolute mode with positive direction of rotation** (possible only for rotary axes).

Program code	Comment
POS[B]=CACN(10)	; Indexing axis B approaches the coded position (index position) 10 in **absolute mode with negative direction of rotation** (possible only for rotary axes).

Program code	Comment
POS[B]=CDC(50)	; Indexing axis B approaches indexing position 50 **directly along the shortest path** (possible only for rotary axes).

Program code	Comment
POS[B]=CIC(-4)	; Indexing axis B traverses four indexing positions **incrementally** from its current position. **in a negative direction**.

Program code	Comment
POS[B]=CIC(35)	; Indexing axis B traverses 35 indexing positions **incrementally** from its present position **in a positive direction**.

Special features

- Modulo rotary axis as indexing axis

  On modulo rotary axes, the indexing positions are divided in factors of 360° and approached directly.

- Indexing axis is between two indexing positions

  The specified position instructions have the following effect in the AUTOMATIC mode.

POS[B]=CIC(1)	The **next higher** indexing position is approached.
POS[B]=CIC(-1)	The **next lower** indexing position is approached.
POS[B]=CIC(0)	The indexing axis is **not** traversed.

Display of indexing position

The number of the indexing position programmed last can be read with the following system variables:

$AA_PROG_INDEX_AX_POS_NO

The number of the indexing position traversed last can be displayed with the following system variables:

$AA_ACT_INDEX_AX_POS_NO

The display depends on the setting in machine data:

MD10940 $MN_INDEX_AX_MODE (settings for indexing position)

Bit	Value	Meaning
0	0	The indexing position changes when the indexing position is reached ("exact stop fine" window) and remains unchanged until the next indexing position is reached. The indexing area thus begins at one indexing position and ends in front of the next indexing position.
	1	The indexing position changes when half the indexing position is reached. A quasi-symmetrical indexing area is thus applied around the indexing position (symmetrical only on linear axes with equidistant indexing or modulo rotary axes on which the indexing area is an integer multiple of the modulo range (MD30330 $MA_MODULO_RANGE), otherwise proportional to the distances between the indexing positions). On **modulo rotary axes**, the area between the last indexing position and the first indexing position is **divided proportionally** based on the lengths of the first indexing area and the last indexing area.

The following graphics will illustrate the difference between Bit 0 = 0 and Bit 0 = 1:

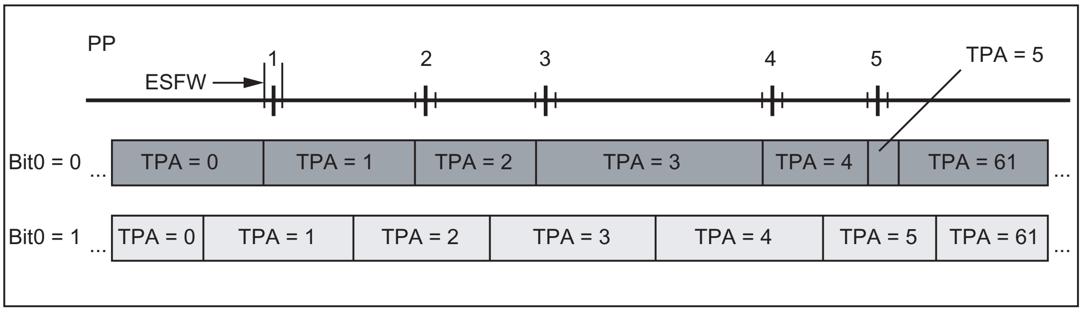
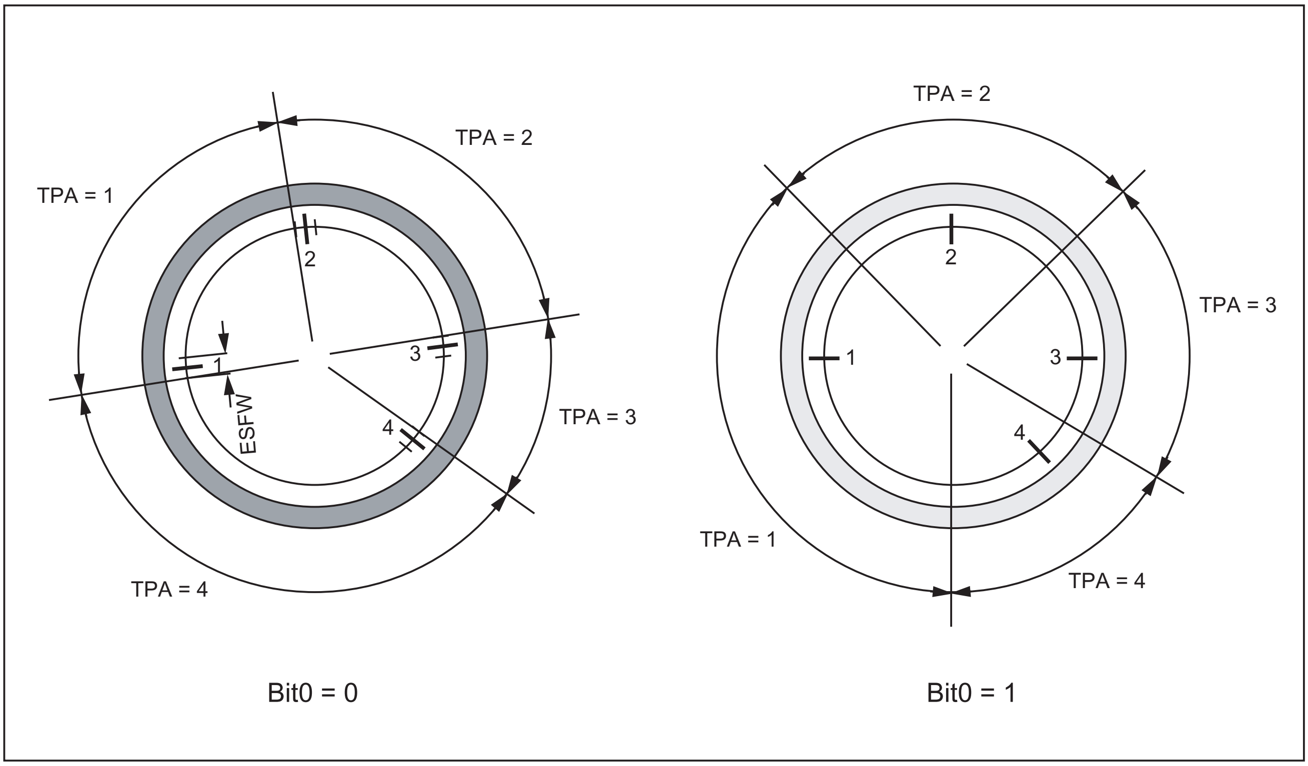


Figure 15-1 Indexing position displays: Linear axis



TP	Programmed indexing position
TPA	Displayed indexing position
ESFW	"Exact stop fine" window

Figure 15-2 Indexing position displays: Modulo rotary axis

Value range of $AA_ACT_INDEX_AX_POS_NO

Expected value ranges of system variables $AA_ACT_INDEX_AX_POS_NO:

Indexing positions from table			
Modulo rotary axis	1 ... n	None 0 n = maximum 60	
Linear axis	0*, 1, 2, 3, ... 59, 60, 61*	0*:	**below** total indexing area
		61*:	**above** total indexing area

Equidistant indexing positions		
Modulo rotary axis	1 ... m	None 0 m = denominator (counter)
Linear axis	... -3, -2, -1, 0, 1, 2, 3, ...	

Traversing to the next indexing position

The response to the "Travel to the next indexing position" command depends on the setting in machine data:

MD10940 $MN_INDEX_AX_MODE (settings for indexing position)

Bit	Value	Meaning
0	0	The next indexing position is approached.
	1	The next indexing position in the direction of motion is always approached.

The following example will serve as explanation:

Bit 0 = 1 and axis below indexing position (but outside "exact stop fine" window).

Although the system variable $AA_ACT_INDEX_AX_POS_NO is indicating indexing position 2, indexing position 2 and **not** indexing position 3 is approached with the "Traverse to next position" command. The next indexing position (in this case indexing position 3) is not approached with the "Traverse to next position" command until the axis is located exactly at (exact stop fine) or above the indexing position.

The nearest indexing position in the current direction of motion is always approached! Under certain circumstances, it is thus necessary to transmit the "traverse to next position" command twice to move from the currently displayed indexing position to the next indexing position number (e.g. from 2 to 3).

FRAMES

Since the control interprets the positions stored in the indexing position table in mm, inches or degrees, FRAMES are not interlocked on indexing axes.

FRAMES are not usually needed for indexing axes, depending on the area of application. It is advisable in most cases to suppress FRAMES and work offsets for indexing axes in the part program.

15.5 Equidistant index intervals

15.5.1 Features

The following exist:

- Any number of equidistant index intervals
- Modified action of MD for indexing axes

Equidistant index intervals can be used for:

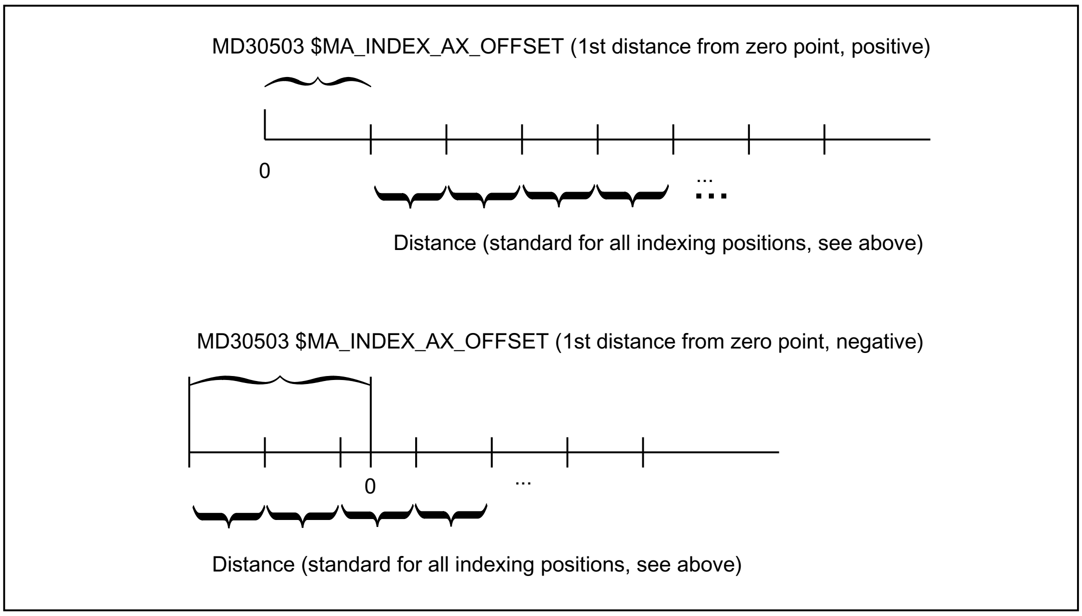
- Linear axes
- Modulo rotary axes
- Rotary axes

Distance between indexes

The index distance is determined for equidistant index intervals according to the following formula:

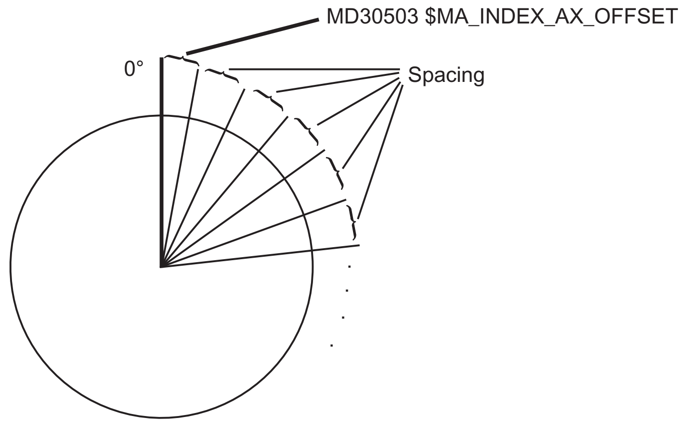
$$\text{Distance} = \frac{\text{Numerator (MD30501 \$MA\_INDEX\_AX\_NUMERATOR)}}{\text{Denominator (MD30502 \$MA\_INDEX\_AX\_DENOMINATOR)}}$$

Linear axis



Modulo rotary axis

$$\text{Index} = \frac{\text{Numerator (MD30330 \$MA\_MODULO\_RANGE)}}{\text{Denominator (MD30502 \$MA\_INDEX\_AX\_DENOMINATOR)}}$$



$$\text{Index} = \frac{\text{Size of modulo range (MD30330 \$MA\_MODULO\_RANGE)}}{\text{Number of indexes in modulo range (MD30502 \$MA\_INDEX\_AX\_DENOMINATOR)}}$$

Activation

The functions with equidistant indexing for an axis (linear axis, modulo rotary axis or rotary axis) is activated in the following settings

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[axis] = 3

15.5.2 Hirth tooth system

Function

With Hirth tooth systems, positions of rotation on a rotary axis are usually interlocked using a latch or other toothed wheel via a linear axis. The interlock should only be activated when an indexing position has been reached precisely. The distance between the indexing positions is the same (equidistant) across the entire circumference.

Requirements

The rotary axis must be an indexing axis. The axis must be referenced.

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

Activation

Machine data:
MD30505 $MA_HIRTH_IS_ACTIVE (axis is an indexing axis with Hirth gearing)
must be set to 1.

Machine data:
MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB (axis is an indexing axis)
must be set to 3 (equidistant indexing positions).

Effect

- The rotary axis can only approach indexing positions in all modes and operating states.
- In JOG mode, the axis can be traversed under JOG control or incrementally.

  Requirement: The axis is referenced.
- Handwheel travel is not possible (see Section "H1: Manual and handwheel travel (Page 143)").
- Only "coded positions" can be approached in AUTO, MDA or via ASUBs.
- The PLC can only move the axis to indexing positions.

15.5.3 Response of the Hirth axes in particular situations

STOP/RESET

For an NC Stop and Reset during a traversing movement, the next indexing position is approached.

Emergency Stop

After an Emergency Stop, the PLC or the operator must move the indexing axis back to an indexing position with JOG before the longitudinal axis can be moved in/down.

Override = 0 or signal "axis stopped"

If the axis has already moved away from the previous indexing position when these events occur, the control moves the axis to the next possible indexing position before the response is initiated.

Delete distance-to-go

After traversing to the next possible indexing position, the movement is aborted at this position.

Command axes

If `MOV=0` is specified for a moving command axis, the axis continues traversing to the next possible indexing position.

References:
Function Manual, Synchronized Actions

MOV command

`MOV=1`	Works on indexing axes with and without Hirth tooth system.
`MOV=0`	Same function for both: Approaches the next position.

DELDTG command

In the case of indexing axes without Hirth tooth system:	Axis stops immediately.
In the case of indexing axes with Hirth tooth system:	Axis traverses to next position.

15.5.4 Restrictions

Transformations

The axis for which the Hirth tooth system is defined cannot take part in kinematic transformations.

PRESET

The axis for which the Hirth tooth system is defined cannot be set to a new value with `PRESET`.

Revolutional feedrate

The axis for which the Hirth tooth system is defined cannot be traversed at revolutional feedrate.

Path/velocity overlay

The axis for which the Hirth tooth system is defined cannot be used with path or velocity overlay.

Frames, ext. work offset, DRF

The axis for which the Hirth tooth system is defined does not support frames or interpolation compensation such as external work offsets, DRF, etc.

Couplings

A Hirth tooth system axis can never be one of the following axis types:

- following axis with master value coupling
- coupled-motion axis
- gantry following axis

References:
Function Manual, Special Functions, Axis Couplings (M3)

15.5.5 Modified activation of machine data

RESET

A RESET is required in order to activate the following machine data after new values have been assigned to them:

MD10900 $MN_INDEX_AX_LENGTH_POS_TAB_1

MD10920 $MN_INDEX_AX_LENGTH_POS_TAB_2

MD10910 $MN_INDEX_AX_POS_TAB_1

MD10930 $MN_INDEX_AX_POS_TAB_2

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB

15.6 Starting up indexing axes

Procedure

The procedure for starting up indexing axes is identical to normal NC axes (linear and rotary axes).

Rotary axis

If the indexing axis is defined as a rotary axis (MD30300 $MA_IS_ROT_AX = "1") with modulo 360° conversion (MD30310 $MA_ROT_IS_MODULO = "1"), indexing positions are also approached with modulo 360°. Only positions within the range from 0° to 359.999° can then be entered in the indexing position table. Otherwise alarm 4080 "Incorrect configuration for indexing axis in MD [Name]" is output during power-up.

The position display can be set to modulo 360° as follows:

MD30320 $MA_DISPLAY_IS_MODULO = 1

Special machine data

The following machine data must be set in addition:

General machine data	
MD10900 $MN_INDEX_AX_LENGTH_POS_TAB_1	Number of positions for indexing axis table 1
MD10920 $MN_INDEX_AX_LENGTH_POS_TAB_2	Number of positions for indexing axis table 2
MD10910 $MN_INDEX_AX_POS_TAB_1 [n]	Indexing position table 1
MD10930 $MN_INDEX_AX_POS_TAB_2 [n]	Indexing position table 2

Axial machine data	
MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB	Axis is indexing axis (assignment of indexing position table 1 or 2, or 3 for equidistant indexing)
MD30505 $MA_HIRTH_IS_ACTIVE	Axis has "Hirth tooth system" property
MD30501 INDEX_AX_NUMERATOR	Numerator for equidistant indexing
MD30502 INDEX_AX_DENOMINATOR	Denominator for equidistant indexing
MD30503 INDEX_AX_OFFSET	Distance of 1st indexing position from zero

Examples

The assignment of the above machine data is described in the following paragraphs using two examples.

Example 1: Indexing axis as rotary axis

Tool turret with 8 locations. The tool turret is defined as a continuously rotating rotary axis. The distances between the 8 turret locations are constant. The first turret location is located at position 0°:

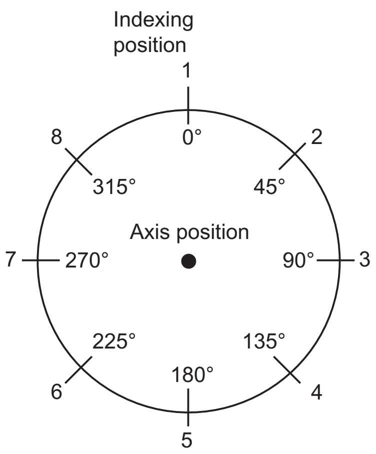


Figure 15-3 Example: Tool turret with 8 locations

The indexing positions for the tool turret are entered in indexing position table 1.

MD10910 $MN_INDEX_AX_POS_TAB_1[0] = 0	; 1. indexing position at 0°
MD10910 $MN_INDEX_AX_POS_TAB_1[1] = 45	; 2. Indexing position at 45°
MD10910 $MN_INDEX_AX_POS_TAB_1[2] = 90	; 3. Indexing position at 90°
MD10910 $MN_INDEX_AX_POS_TAB_1[3] = 135	; 4. Indexing position at 135°
MD10910 $MN_INDEX_AX_POS_TAB_1[4] = 180	; 5. Indexing position at 180°
MD10910 $MN_INDEX_AX_POS_TAB_1[5] = 225	; 6. Indexing position at 225°
MD10910 $MN_INDEX_AX_POS_TAB_1[6] = 270	; 7. Indexing position at 270°
MD10910 $MN_INDEX_AX_POS_TAB_1[7] = 315	; 8. Indexing position at 315°

Other machine data:

MD10900 $MN_INDEX_AX_LENGTH_POS_TAB_1= 8	; 8 indexing positions in table 1
MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB [AX5] = 1	;; Axis 5 is defined as indexing axis Indexing positions in table 1
MD30300 $MA_IS_ROT_AX [AX5] = 1	; Axis 5 is rotary axis
MD30310 $MA_ ROT_IS_MODULO [AX5] = 1	; Modulo conversion is activated

Example 2: Indexing axis as linear axis

Workholder with 10 locations.

The distances between the 10 locations are different. The first location is at position -100 mm.

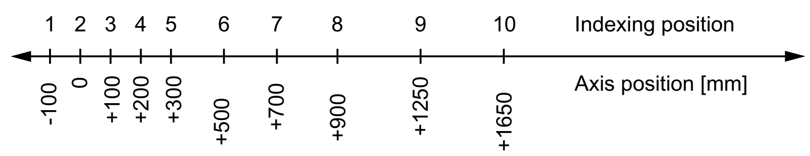


Figure 15-4 Example: Workholder as an indexing axis

The indexing positions for the workholder are entered in table 2:

MD10930 $MN_INDEX_AX_POS_TAB_2[0] = -100	; 1. indexing position at -100
MD10930 $MN_INDEX_AX_POS_TAB_2[1] = 0	; 2. indexing position at 0
MD10930 $MN_INDEX_AX_POS_TAB_2[2] = 100	; 3. indexing position at 100
MD10930 $MN_INDEX_AX_POS_TAB_2[3] = 200	; 4. indexing position at 200
MD10930 $MN_INDEX_AX_POS_TAB_2[4] = 300	; 5. indexing position at 300
MD10930 $MN_INDEX_AX_POS_TAB_2[5] = 500	; 6. indexing position at 500
MD10930 $MN_INDEX_AX_POS_TAB_2[6] = 700	; 7. indexing position at 700
MD10930 $MN_INDEX_AX_POS_TAB_2[7] = 900	; 8. indexing position at 900
MD10930 $MN_INDEX_AX_POS_TAB_2[8] = 1250	; 9. indexing position at 1250
MD10930 $MN_INDEX_AX_POS_TAB_2[9] = 1650	; 10. indexing position at 1650

Other machine data:

MD10920 $MN_INDEX_AX_LENGTH_POS_TAB_2=10	; 10 indexing positions in table 2
MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB [AX6] = 2;	Axis 6 is defined as indexing axis, indexing positions in table 2

15.7 Special features of indexing axes

DRF

An additional incremental work offset can also be generated for indexing axes in AUTOMATIC mode with the handwheel using the `DRF` function.

Software limit switch

The software limit switches are also effective during traversing movements once the indexing axis has been referenced.

In handwheel traversing with JOG continuous or JOG incremental, the indexing axis stops at the indexing position ahead of the software limit switch.

Reference point approach

When reference point is reached::
DB31, ... DBX60.4 or 60.5 (referenced/synchronized 1 or 2) = 1
the indexing axis moves only to indexing positions in JOG continuous and JOG incremental mode.

If the axis is not referenced:
DB31, ... DBX60.4 or 60.5 (referenced/synchronized 1 or 2) = 0,
the indexing positions are ignored when traversing in JOG mode!

Since the axis positions stored in the indexing position tables only correspond to the machine positions when the axis is referenced, an NC start must be disabled for as long as the indexing axis is not referenced.

Position display

Positions on indexing axes are displayed in the units of measurement normally used for the axes (mm, inches or degrees).

Abort through RESET

`RESET` causes the traversing movement on an indexing axis to be aborted and the axis to be stopped. The indexing axis is no longer positioned on an indexing position.

15.8 Examples

15.8.1 Examples of equidistant indexes

Modulo rotary axis

MD30502 $MA_INDEX_AX_DENOMINATOR[AX4] = 18

MD30503 $MA_INDEX_AX_OFFSET[AX4] = 5

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3

MD30300 $MA_IS_ROT_AX[AX4] = TRUE

MD30310 $MA_ROT_IS_MODULO[AX4] = TRUE

With the machine data above, axis 4 is defined as a modulo rotary axis and an indexing axis with equidistant positions every 20° starting at 5°.
The following indexing positions result:
5, 25, 45, 65, 85, 105, 125, 145, 165, 185, 205, 225, 245, 265, 285, 305, 325 and 245 degrees.

Note

The assignment:
MD30502 $MA_INDEX_AX_DENOMINATOR[AX4] =18
results in a 20° division because the default for machine data MD30330 $MA_MODULO_RANGE
is 360°.

Rotary axis

MD30501 $MA_INDEX_AX_NUMERATOR[AX4] = 360

MD30502 $MA_INDEX_AX_DENOMINATOR[AX4] = 18

MD30503 $MA_INDEX_AX_OFFSET[AX4] = 100

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3

MD30300 $MA_IS_ROT_AX[AX4] = TRUE

MD36100 $MA_POS_LIMIT_MINUS[AX1] = 100

MD36110 $MA_POS_LIMIT_PLUS[AX1] = 260

With the machine data above, axis 4 is defined as a rotary axis and an indexing axis with equidistant positions every 20° starting at 100°.
The following indexing positions result:
100°, 120°, 140° etc.
Positions less than 100° cannot be approached as indexing positions.
It is advisable to place the lower software limit switch in this case. The indexing positions continue until the software limit switch is reached (in this case 260°). The rotary axis can therefore only traverse between 100° and 260°.

Linear axis

MD30501 $MA_INDEX_AX_NUMERATOR[AX1] = 10

MD30502 $MA_INDEX_AX_DENOMINATOR[AX1] = 1

MD30503 $MA_INDEX_AX_OFFSET[AX1] = -200

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[AX1] = 3

MD30300 $MA_IS_ROT_AX[AX1] = FALSE

MD36100 $MA_POS_LIMIT_MINUS[AX1] = -200

MD36110 $MA_POS_LIMIT_PLUS[AX1] = 200

With the machine data above, axis 4 is defined as a linear axis and an indexing axis with equidistant positions every 10 mm starting at -200 mm.
The following indexing positions result:
-200, -190, -180 mm etc.
These indexing positions continue until the software limit switch is reached (in this case 200 mm).

Hirth tooth system

MD30502 $MA_INDEX_AX_DENOMINATOR[AX4] = 360

MD30503 $MA_INDEX_AX_OFFSET[AX4] = 0

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3

MD30300 $MA_IS_ROT_AX[AX4] = TRUE

MD30310 $MA_ROT_IS_MODULO[AX5] = TRUE

MD30505 $MA_HIRTH_IS_ACTIVE[AX4] = TRUE

With the machine data above, axis 4 is defined as a modulo rotary axis and an indexing axis with Hirth tooth system and equidistant positions every 1° starting at 0°.

15.9 Data lists

15.9.1 Machine data

15.9.1.1 General machine data

Number	Identifier: $MN_	Description
10260	CONVERT_SCALING_SYSTEM	Basic system switchover active
10270	POS_TAB_SCALING_SYSTEM	System of measurement of position tables
10900	INDEX_AX_LENGTH_POS_TAB_1	Number of positions for indexing axis table 1
10910	INDEX_AX_POS_TAB_1[n]	Indexing position table 1
10920	INDEX_AX_LENGTH_POS_TAB_2	Number of positions for indexing axis table 2
10930	INDEX_AX_POS_TAB_2[n]	Indexing position table 2
10940	INDEX_AX_MODE	Options for indexing positions

15.9.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis
30310	ROT_IS_MODULO	Modulo conversion for rotary axis
30320	DISPLAY_IS_MODULO	Position display is modulo 360°
30500	INDEX_AX_ASSIGN_POS_TAB	Axis is indexing axis
30501	INDEX_AX_NUMERATOR	Numerator for indexing axes with equidistant positions
30502	INDEX_AX_DENOMINATOR	Denominator for indexing axes with equidistant positions
30503	INDEX_AX_OFFSET	Indexing position for indexing axes with equidistant positions
30505	HIRTH_IS_ACTIVE	Hirth tooth system is active

15.9.2 Setting data

15.9.2.1 General setting data

Number	Identifier: $SN_	Description
41050	JOG_CONT_MODE_LEVELTRIGGRD	JOG continuous in inching mode

15.9.3 Signals

15.9.3.1 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized 1, referenced/synchronized 2	DB31, ...DBX60.4/5	DB390x.DBX0.4/5
Indexing axis in position	DB31, ...DBX76.6	DB390x.DBX1002.6

15.9.4 System variables

Identifier	Description
$AA_ACT_INDEX_AX_POS_NO[axis]	Number of last indexing position reached or overtraveled
$AA_PROG_INDEX_AX_POS_NO[axis]	Number of programmed indexing position

W3: Tool change 16

16.1 Brief Description

Tool change

CNC-controlled machine tools are equipped with tool magazines and automatic tool change facility for the complete machining of workpieces.

Sequence

The procedure for changing tools comprises three steps:

1. Movement of the tool carrier from the machining position to the tool change position
2. Tool change
3. Movement of the tool carrier from the tool change position to the new machining position.

Requirements

The following is required for tool change:

- short idle times
- Time-saving searches, provision and return of tool during the machining time.
- Simple programming of the tool change cycle
- Automatic flow of the required axis and gripper movements
- Easy fault recovery

16.2 Tool magazines and tool change equipments

Tool magazines and tool changing equipment are selected according to the machine type:

Machine type	Tool magazine	Tool change equipment
Turning machines	Turret (disk, flat, inclined)	No special tool change equipment. The tool is changed by turning the turret
Milling machines	Magazines (chain, disk-type, rotary-plate, cartridge)	Gripper/dual gripper as tool change equipment.

As the changing operation interrupts the machining, idle times times must be minimized.

16.3 Tool change times

Tool change times depend strongly on the design layout of the machine tool.

Typical tool change times	
0.1 to 0.2 s	for advancing a turret
0.3 to 2 s	for tool change with gripper for a prepared tool

16.4 Cut-to-cut time

The cut-to-cut time is the period that elapses when a tool is changed between retraction from the interruption point on the contour (from cut) and repositioning at the interruption point (return to cut) with the new tool when the spindle is rotating.

Typical cut-to-cut times are as follows:

Typical cut-to-cut times	
0.3 to 1 s	for turning machine with turret
0.5 to 5 s	for milling machine with automatic tool changer

16.5 Starting the tool change

Variants

The tool change can be actuated by:

- T function
- M command (preferably M06)

Parameterization

Which control versions should be effective is defined with the machine data:

MD22550 $MC_TOOL_CHANGE_MODE

Value	Meaning	Typical application
0	The T function loads the new tool immediately.	Turning machines with tool turret
1	New tool with T function prepared for change and placed in the tool change position simultaneously during the machining time. The M command is used to remove the old tool from the spindle and load the new tool.	Milling machines with a tool magazine,

The M command for tool change is defined in machine data:

MD22560 $MC_TOOL_CHANGE_M_CODE

Default setting is 6 (corresponding to DIN 66025).

Note

If the tool offset number is supplied from the PLC or an HMI tool manager, a preprocessing stop STOPRE must be inserted at a suitable point. STOPRE must be avoided, however, when tool radius compensation (G41 / G42) or SPLINE interpolation is active, since several blocks are required here in advance for the path calculation.

References

For further information about M functions which also apply to tool change M06 (e.g. extended address, time of output to PLC, auxiliary function groups, behavior during block search, behavior during overstore) see:

Function Manual, Synchronized Actions

16.6 Tool change point

Tool change point

The selection of the tool change point has a significant effect on the cut-to-cut time (Page 796). The tool change point is chosen according to the machine tool concept and, in certain cases, according to the current machining task.

Approaching a fixed point

Fixed positions on a machine axis stored in machine data can be approached by means of the "fixed-point approach" function. This can be used to define and control one or several tool change points.

There are two fixed point approach options:

- Approaching a fixed point in JOG

  The machine user starts the "fixed-point approach" in the JOG mode with the traverse keys or the handwheel (see Section "Approaching a fixed point in JOG (Page 183)").

- Approaching a fixed point with G75

  Fixed point approach is called using the command `G75` from the part program.

  References:
  Programming Manual, Fundamentals, Section: Additional commands > Approach fixed point (G75)

16.7 Supplementary Conditions

The tool change requires, amongst other things, a tool management system which ensures that the tool to be loaded is available at the tool change position at the right time.

16.8 Examples

Milling machine

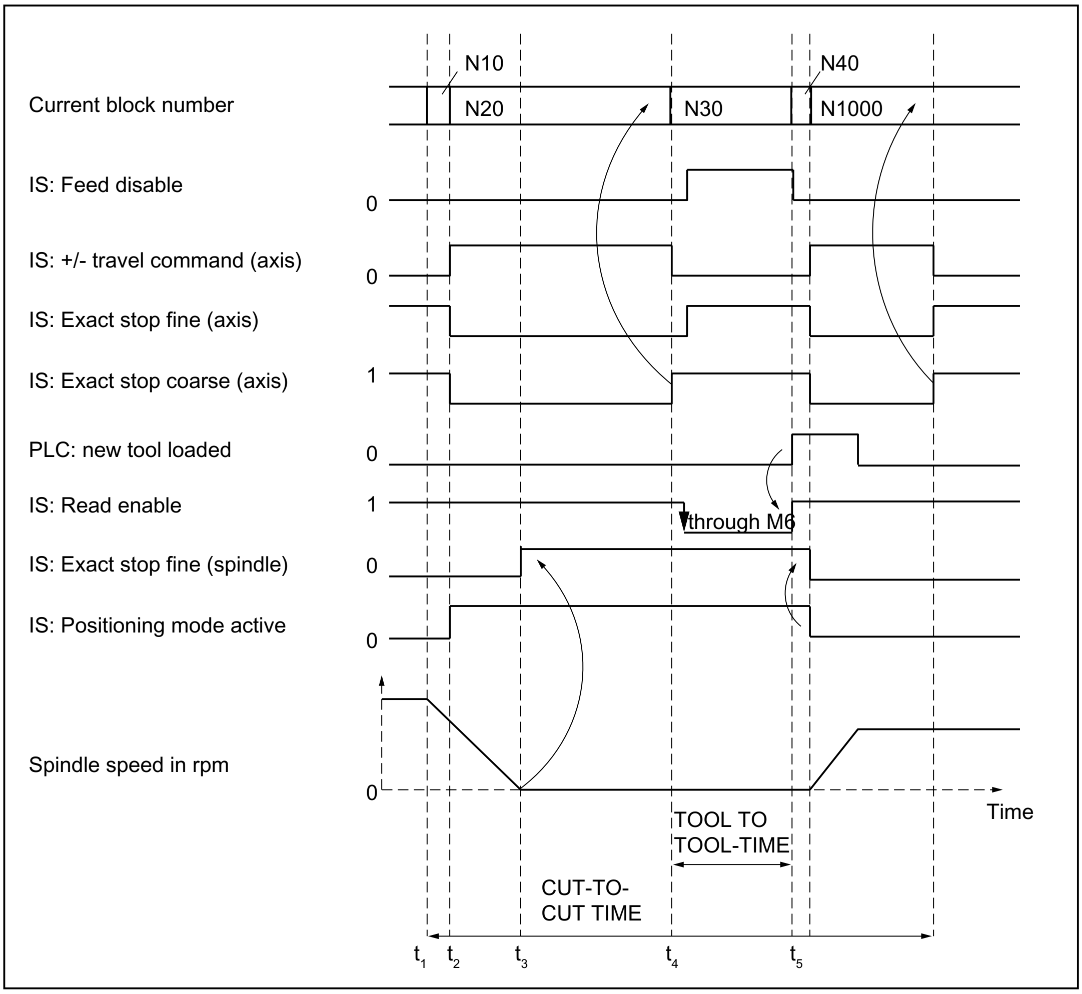
The following example shows a typical cut-to-cut sequence of operations for a tool change with a tool changer and a fixed absolute tool change point on a milling machine.

Machining program:

Program code	Comment
N970 G0 X=... Y=... Z=... LF	; Retraction from contour
N980 T1 LF	; Tool preselection
N990 W_WECHSEL LF	; Subroutine call without parameters
N1000 G90 G0 X=... Y=... Z=... M3 S1000 LF	; Continue machining

Subroutine for tool change:

Program code	Comment
PROC W_WECHSEL LF	
N10 SPOSA=... S0 LF	; Spindle positioning
N20 G75 FP=2 X1=0 Y1=0 Z1=0	; Approach tool change position
N30 M06 LF	; Change tool
N40 M17 LF	



t_1: Axes stationary.
Spindle rotates.
Start of tool change cycle in `N10`.

t_2: Move axes to tool change point with `G75` in `N20`.

t_3: Spindle reaches programmed position from block `N10`.

t_4: Axes reach exact stop coarse from `N20`; `N30` thus begins:
`M06` removes the previous tool from the spindle and loads and clamps the new tool.

t_5: Tool changer swivels back to original position.

Figure 16-1 Chronological sequence of tool change

Then, in `N1000` of the calling main program:

- The new tool offset can be selected
- the axes can be returned to the contour, or
- the spindle can be accelerated.

16.9 Data lists

16.9.1 Machine data

16.9.1.1 General machine data

Number	Identifier: $MN_	Description
18082	MM_NUM_TOOL	Number of tools

16.9.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
22200	AUXFU_M_SYNC_TYPE	Output timing of M functions
22220	AUXFU_T_SYNC_TYPE	Output timing of T functions
22550	TOOL_CHANGE_MODE	New tool offset for M function
22560	TOOL_CHANGE_M_CODE	M function for tool change

16.9.1.3 Axis-/spindlespecific machine data

Number	Identifier: $MA_	Description
30600	FIX_POINT_POS[n].	Fixed point positions of the machine axes for G75

16.9.2 Signals

16.9.2.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
M function M06	DB21, ...DBX194.6	DB2500.DBB1000.6

W4: Grinding-specific tool offset and tool monitoring 17

Contents

The topics of this functional description are:

- Grinding-specific tool offset
- Online tool offsets (continuous dressing)
- Grinding-specific tool monitoring
- Constant grinding wheel peripheral speed (GWPS)

References

For fundamentals see:

- Function Manual Basic Functions; Tool Offset (W1)

Programming, mode of operation and handling, please refer to:

- Programming Manual, Fundamentals

17.1 Tool offset for grinding operations

17.1.1 Structure of tool data

Grinding tools

Grinding tools are tools of types 400 to 499.

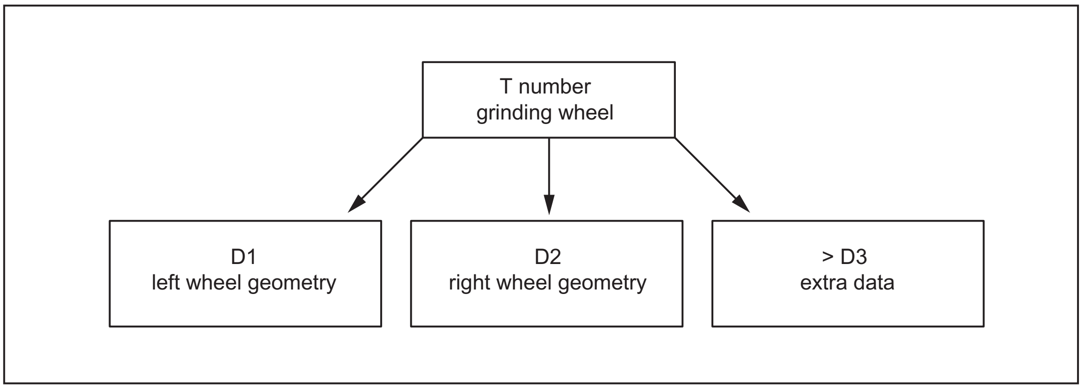
Tool offset for grinding tools

Grinding tools normally have specific tool and dresser data in addition to cutting edge data.

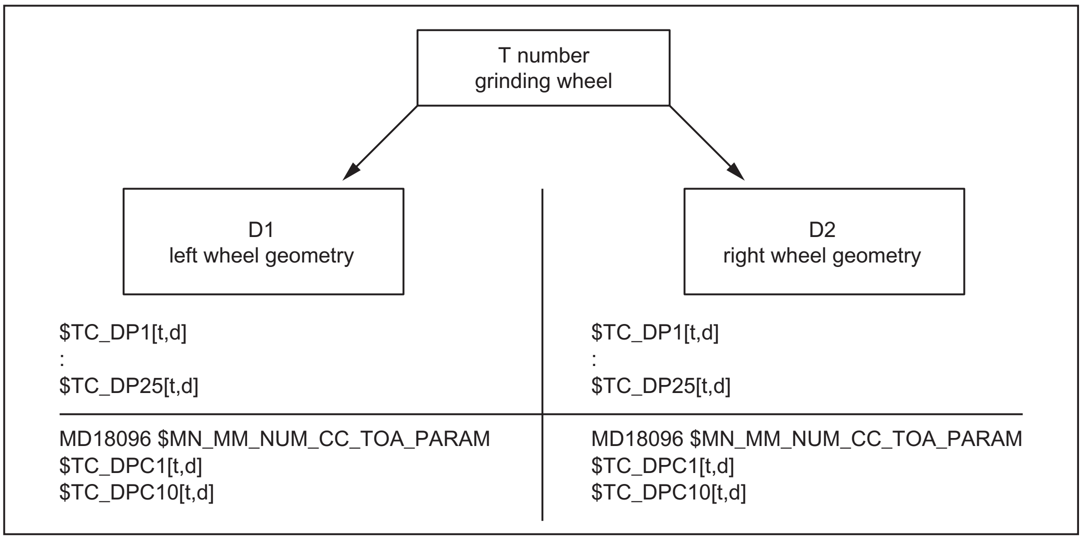
The specific grinding wheel data for the left and right wheel geometry can be stored under a T number in `D1` and `D2`.

If data are needed for the dresser geometry, they can be stored, e.g., starting at `D3` of a T number or in additional cutting-edge-specific data
(MD18096 $MN_MM_NUM_CC_TOA_PARAM).

Example 1:



Example 2:



All offsets belonging to a grinding wheel and dresser can be combined in the tool edges `D1` and `D2` for the grinding wheel and, for example, `D3` and `D4` for the dresser:

- `D1`: grinding wheel geometry left
- `D2`: Grinding wheel geometry right
- `D3`: Dresser geometry left
- `D4`: Dresser geometry right

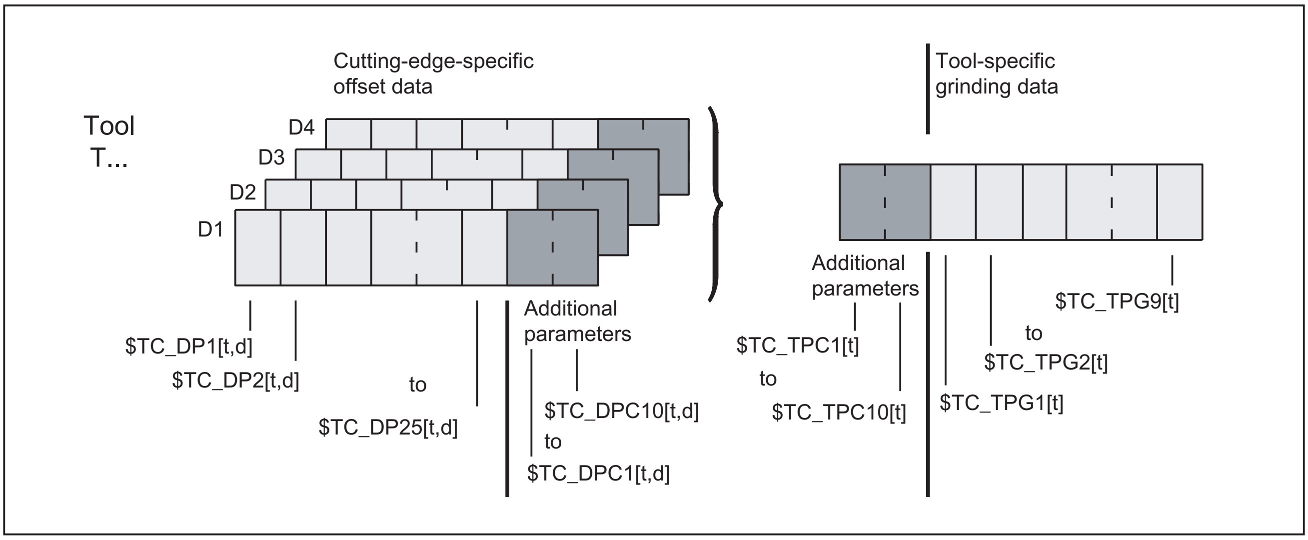


Figure 17-1 Structure of tool offset data for grinding tools

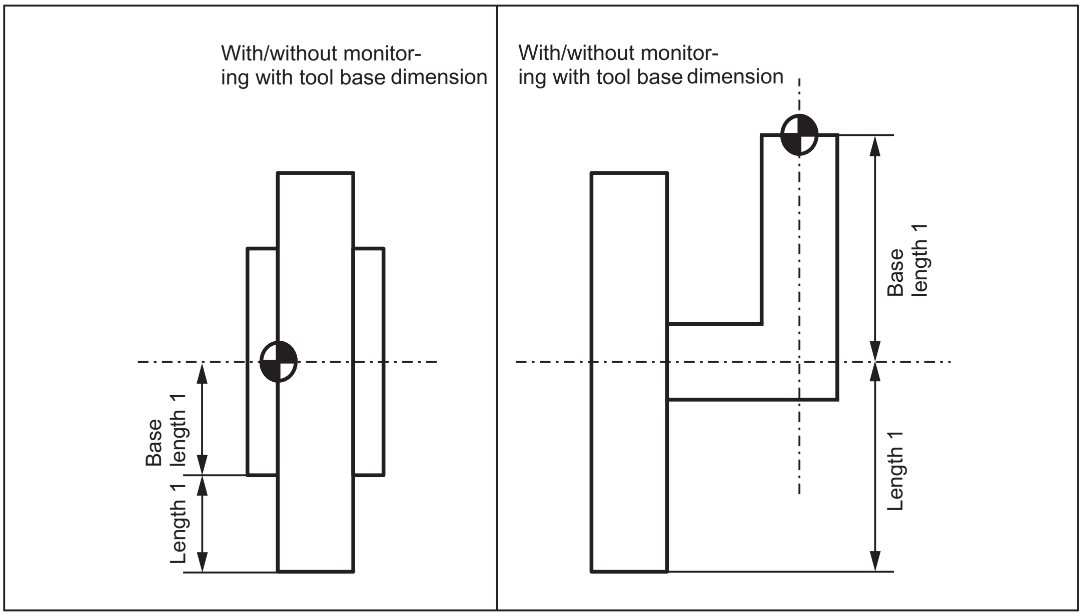
17.1.2 Edge-specific offset data

Tool parameter

The tool parameters for grinding tools have the same meaning as those for turning and milling tools.

Tool parameter	Meaning	Comment	
1	Tool type		
2	Cutting edge position	For turning and grinding tools only	
Geometry tool length compensation			
3	Length 1		
4	Length 2		
5	Length 3		
Geometry tool radius compensation			
6	Radius 1		
7			Reserved
8			Reserved
9			Reserved
10			Reserved
11			Reserved
Wear tool length compensation			
12	Length 1		
13	Length 2		
14	Length 3		

Tool parameter	Meaning	Comment	
Wear tool radius compensation			
15	Radius 1		
16			Reserved
17			Reserved
18			Reserved
19			Reserved
20			Reserved
Tool base dimension / adapter dimension tool length compensation			
21	Basic length 1		
22	Basic length 2		
23	Basic length 3		
Technology			
24	Undercut angle	For turning tools only	
25			Reserved



Note

The cutting edge data for `D1` and `D2` of a selected grinding tool can be chained, i.e. if a parameter in `D1` or `D2` is modified, then the same parameter in `D1` or `D2` is automatically overwritten with the new value (see tool-specific data $TC_TPG2).

Definition of additional parameters $TC_DPC1...10

For user-specific cutting edge data, additional parameters $TC_DPC1 to 10 can be set up independent of the tool type using the following general machine data:

MD18096 $MN_MM_NUM_CC_TOA_PARAM

> ⚠ **CAUTION**
>
> **Data loss**
>
> Changes to MD18096 take effect after power on and will lead to initialization of the memory.
>
> A data backup must be created beforehand.

> ⚠ **CAUTION**
>
> **No grinding wheel offset changeover**
>
> Automatic changeover between grinding wheel offset left and right does **not** take place during contour grinding.
>
> A changeover must be programmed.

Tool types for grinding tools

The structure of tool types for grinding tools is as follows:

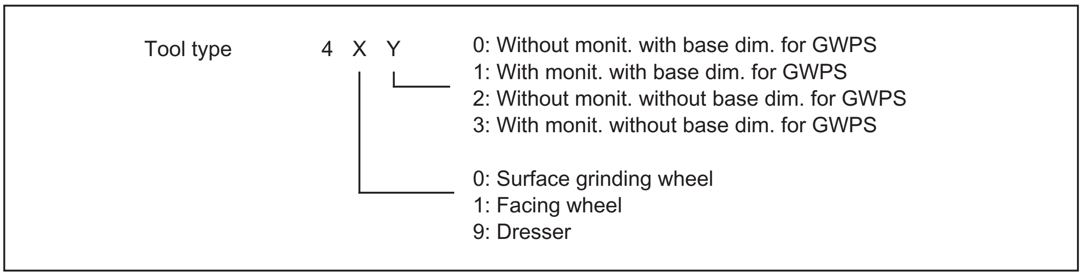


Figure 17-2 Structure of tool type for grinding tools

Note

MD20350 $MC_TOOL_GRIND_AUTO_TMON

Through this channel-specific machine data it can be determined, whether for grinding tools **with monitoring** (i.e. uneven tool types) the monitoring is already active or not when this tool is selected.

This structure can be used to create the following tool types:

Type	Description
400	Surface grinding wheel
401	Surface grinding wheel with monitoring with tool base dimension for GWPS
402	Surface grinding wheel without monitoring without tool base dimension for GWPS
403	Surface grinding wheel with monitoring without tool base dimension for GWPS
410	Facing wheel
411	Facing wheel with monitoring with tool base dimension for GWPS
412	Facing grinding wheel without monitoring without tool base dimension for GWPS
413	Facing wheel with monitoring without tool base dimension for GWPS
490	Dresser

17.1.3 Tool-specific grinding data

Tool-specific grinding data

Tool-specific grinding data is available once for every T number (type 400- 499). The data is automatically set up with every new grinding tool (type 400 - 499).

Note

Tool-specific grinding data has the same characteristics as a tool edge.

This is to be taken into account when the number of cuts is specified:

MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA

When all the cutting edges of a tool are deleted, the existing tool-specific grinding data is deleted at the same time.

Parameters

The parameters are assigned as follows:

Parameter	Meaning	Data type
$TC_TPG1	Spindle number	Integer
$TC_TPG2	Chaining rule	Integer
$TC_TPG3	Minimum wheel radius	Real
$TC_TPG4	Minimum wheel width	Real
$TC_TPG5	Current wheel width	Real
$TC_TPG6	Maximum speed	Real
$TC_TPG7	Maximum peripheral speed	Real
$TC_TPG8	Angle of inclined wheel	Real
$TC_TPG9	Parameter number for radius calculation	Integer

Parameter	Meaning	Data type
Additional parameters (user-specific cutting edge data)		
$TC_TPC1 to $TC_TPC10		Real

Definition of additional parameters $TC_DPC1...10

For the user-specific cutting data the additional parameters $TC_DPC1 to $TC_DPC10 can be implemented independent of the WZ-type. This is done via the general machine data:

MD18096 $MN_MM_NUM_CC_TDA_PARAM

> **CAUTION**
>
> **Data loss**
>
> Changes to MD18096 take effect after power on and will lead to initialization of the memory.
>
> A data backup must be created beforehand.

Spindle number $TC_TPG1

Number of the programmed spindle (e.g. grinding wheel peripheral speed) and spindle to be monitored (e.g. wheel radius and width)

Chain rule $TC_TPG2

This parameter is set to define which tool parameters of tool edge 2 (`D2`) and tool edge 1 (`D1`) have to be chained to one another. When the setpoint of a chained parameter is modified, the value of the parameter with which it is chained is modified automatically.

Tool parameter	Meaning	Bit in $TC_TPG2	Hex	Dec
$TC_DP1	Tool type	0	0001	1
$TC_DP2	Length of cutting edge	1	0002	2
Geometry tool length compensation				
$TC_DP3	Length 1	2	0004	8
$TC_DP4	Length 2	3	0008	16
$TC_DP5	Length 3	4	0010	32
$TC_DP6	Radius	5	0020	64
$TC_DP7	Reserved	6	0040	128
$TC_DP8		7	0080	256
$TC_DP9		8	0100	512
$TC_DP10		9	0200	1024
$TC_DP11	Reserved	10	0400	2048

Tool parameter	Meaning	Bit in $TC_TPG2	Hex	Dec
Wear tool length compensation				
$TC_DP12	Length 1	11	0800	4096
$TC_DP13	Length 2	12	1000	8192
$TC_DP14	Length 3	13	2000	16384
$TC_DP15	Radius	14	4000	32768
$TC_DP16	Reserved	15	8000	65536
$TC_DP17		16	10000	131072
$TC_DP18		17	20000	262144
$TC_DP19		18	40000	524288
$TC_DP20	Reserved	19	80000	1048576
Tool base dimension / adapter dimension tool length compensation				
$TC_DP21	Basic length 1	20	100000	2097152
$TC_DP22	Basic length 2	21	200000	4194304
$TC_DP23	Basic length 3	22	400000	8388608
Technology				
$TC_DP24	Reserved	23	800000	16777216
$TC_DP25	Reserved	24	1000000	33554432

Example of parameter chain:

Lengths 1, 2 and 3 of the geometry, the length wear and the tool base / adapter dimensions of lengths 1, 2 and 3 on a grinding tool (T1 in the example) must be automatically transferred.

Furthermore, the same tool type applies to tool edges 1 and 2.

Tool type	$TC_DP1	Bit 0
Length 1	$TC_DP3	Bit 2
Length 2	$TC_DP4	Bit 3
Length 3	$TC_DP5	Bit 4

Wear

Length 1	$TC_DP12	Bit 11
Length 2	$TC_DP13	Bit 12
Length 3	$TC_DP14	Bit 13

Base/adapter dimension

Length 1	$TC_DP21	Bit 20
Length 2	$TC_DP22	Bit 21
Length 3	$TC_DP23	Bit 22

Parameter $TC_TPG2 must therefore be assigned as follows:

Binary:	$TC_TPG2[1]= 'B111 0000 0011 1000 0001 1101' (Bit 22 ... Bit 0)
Hexadecimal:	$TC_TPG2[1]= 'H70381D'
Decimal:	$TC_TPG2[1]='D7354397'

Note

If the chaining specification is subsequently altered, the values of the two cutting edges are not automatically adjusted, but only after one parameter has been altered.

Minimum wheel radius and width $TC_TPG3 $TC_TPG4

The limit values for the grinding wheel radius and width must be entered in these parameters. These parameter values are used to monitor the grinding wheel geometry.

Note

It must be noted that the minimum grinding wheel radius must be specified in the Cartesian coordinate system for an inclined grinding wheel. A signal is output at the PLC interface if the grinding wheel width and radius drop below the minimum limits. Users can use these signals to define their error strategy.

Current width $TC_TPG5

The width of the grinding wheel measured, for example, after the dressing operation, is entered here.

Maximum speed and grinding wheel peripheral speed $TC_TPG6 $TC_TPG7

The upper limit values for maximum speed and peripheral speed of the grinding wheel must be entered in these parameters.

Precondition: A spindle has been declared.

Angle of inclined wheel $TC_TPG8

This parameter specifies the angle of inclination of an inclined wheel in the current plane. It is evaluated for GWPS.

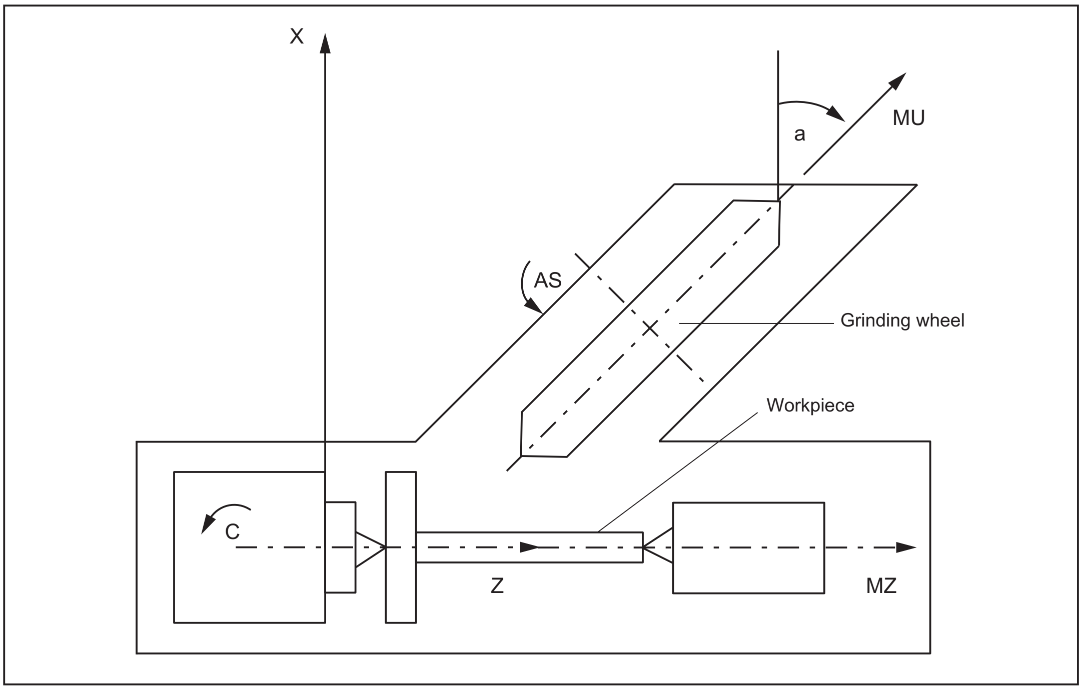


Figure 17-3 Machine with inclined infeed axis

Note

The tool lengths are not automatically compensated when the angle is altered.

The angle must be within the range -90° ≤ $TC_TPG8 < +90°.

On inclined axis machines the same angle must be specified for the inclined axis and the inclined wheel.

Parameter number for radius calculation $TC_TPG9

This parameter specifies which offset values are used for the GWPS calculation and tool monitoring of the minimum wheel radius ($TC_TPG3).

$TC_TPG9 = 3	Length 1 (geometry + wear + base, depending on tool type)
$TC_TPG9 = 4	Length 2 (geometry + wear + base, depending on tool type)
$TC_TPG9 = 5	Length 3 (geometry + wear + base, depending on tool type)
$TC_TPG9 = 6	Radius

Access from part program

Parameters can be read and written from the part program.

Example	Programming
Read the current width of tool 2 and store in R10	`R10 = $TC_TPG5 [2]`
Write value 2000 to the maximum speed of tool 3	`$TC_TPG6 [3] = 2000`

$P_ATPG[m] for current tool

This system variable allows the tool-specific grinding data for the **current** tool to be accessed.

m: Parameter number (data type: Real)

Example:

Parameter 3 ($TPG3[<T No.>])

$P_ATPG[3]=R10

Note

The monitoring data applies to both the left-hand and the right-hand cutting edge of the grinding wheel.

The tool-specific grinding data is activated when `GWPSON` (select constant grinding wheel surface speed) and `TMON` (select tool monitoring) are programmed. To activate a data item that has been modified, it is necessary to program `GWPSON` or `TMON` again.

The length compensations always specify the distances between the tool carrier reference point and the tool tip in the Cartesian coordinates (must be noted for inclined grinding wheel).

17.1.4 Examples of grinding tools

Assignment of length offsets

Tool length compensations for the geometry axes or radius compensation in the plane are assigned on the basis of the current plane.

Planes

The following planes and axis assignments are possible (abscissa, ordinate, applicate for 1st, 2nd and 3rd geometry axes):

Command	Plane (abscissa/ordinate)	Axis perpendicular to plane (applicate)
G17	X/Y	Z
G18	Z/X	Y
G19	Y/Z	X

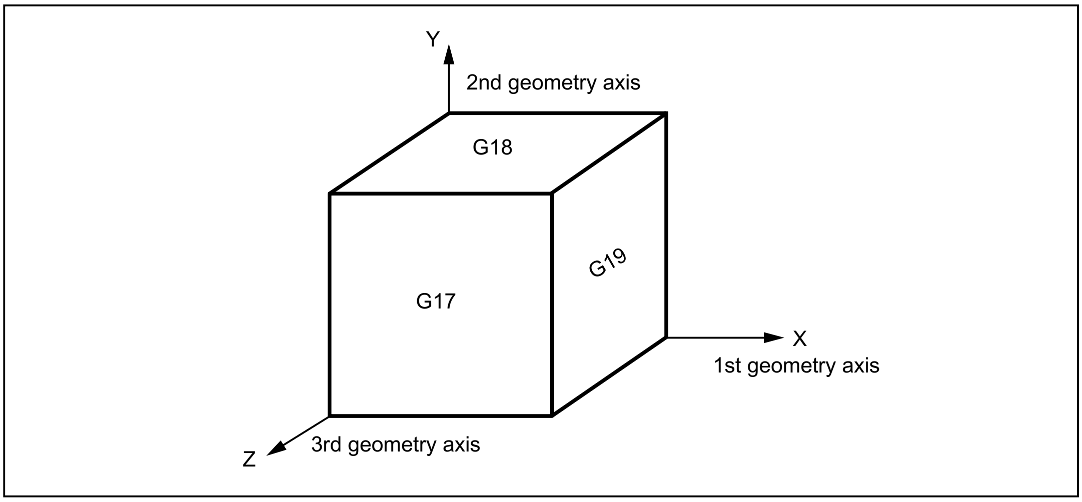


Figure 17-4 Planes and axis assignment

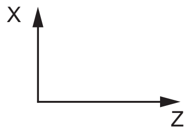
Surface grinding wheel

Entries in tool parameters

$TC_DP1	400
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius

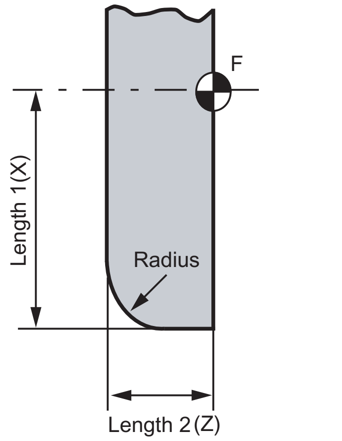
Unused parameters must be set to 0

e.g.
G18: Z/X plane



Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y Radius in Y/Z



F: Tool holder reference point

Figure 17-5 Offset values required by a surface grinding wheel

Inclined wheel

without tool base dimension for GWPS

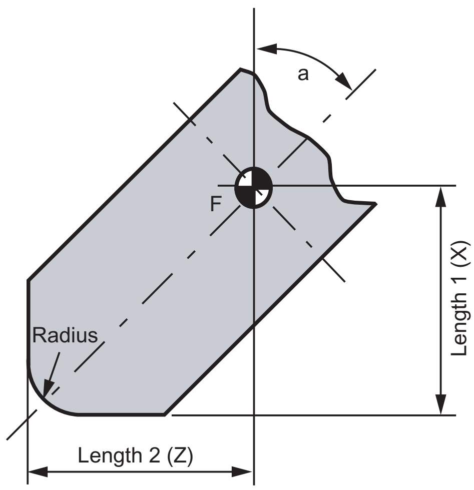
Entries in tool parameters

$TC_DP1	403
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius

Unused parameters must be set to 0

Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y Radius in Y/Z



F: Tool holder reference point

Figure 17-6 Offset values required for inclined wheel with implicit monitoring selection

Inclined wheel

with tool base dimension for GWPS

Entries in tool parameters

$TC_DP1	401
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius
$TC_DP21	L1 base
$TC_DP22	L2 base

Wear values acc. to requirements

Unused parameters must be set to 0

Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y Radius in Y/Z

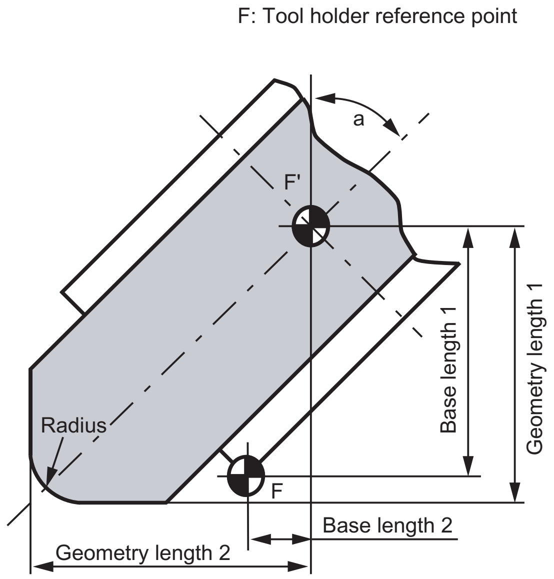


Figure 17-7 Required offset values shown by example of inclined grinding wheel with implicit monitoring selection and with base selection for GWPS calculation

Surface grinding wheel

Entries in tool parameters

$TC_DP1	403
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius
$TC_DP21	L1 base
$TC_DP22	L2 base

Wear values acc. to requirements

Unused parameters must be set to 0

Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y Radius in Y/Z

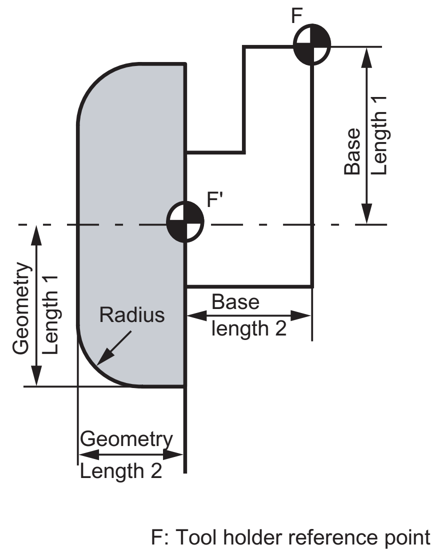


Figure 17-8 Required offset values of a surface grinding wheel without base dimension for GWPS

Facing wheel

Entries in tool parameters

$TC_DP1	411
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius

Wear values acc. to requirements

Unused parameters must be set to 0

Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y/Z

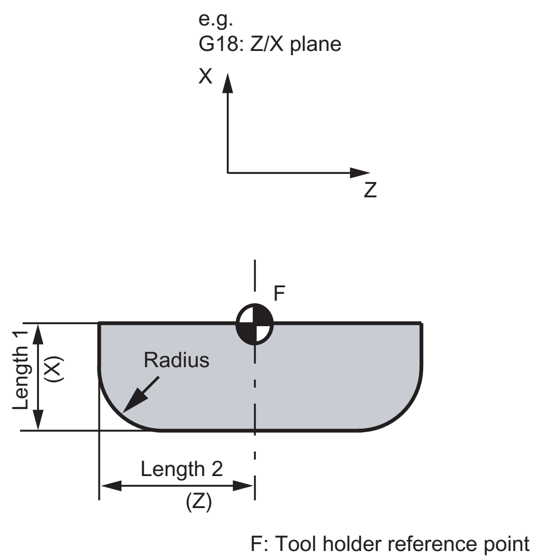


Figure 17-9 Required offset values of a facing wheel with monitoring parameters

17.2 Online tool offset

17.2.1 General information

Application

A grinding operation involves both machining of a workpiece and dressing of the grinding wheel. These processes can take place in the same channel or in separate channels.

To allow the wheel to be dressed while it is machining a workpiece, the machine must offer a function whereby the reduction in the size of the grinding wheel caused by dressing is compensated on the workpiece. This type of compensation can be implemented by means of the "Online tool offset" (Continuous Dressing) function.

Dressing during machining process

To allow machining to continue while the grinding wheel is being dressed, the reduction in the size of the grinding wheel caused by dressing must be transferred to the current tool in the machining channel as a tool offset that is applied immediately.

This parallel dressing operation can be implemented by means of the "Continuous Dressing (parallel dressing), Online tool offset" function.

Note

The online tool offset may only be used for grinding tools.

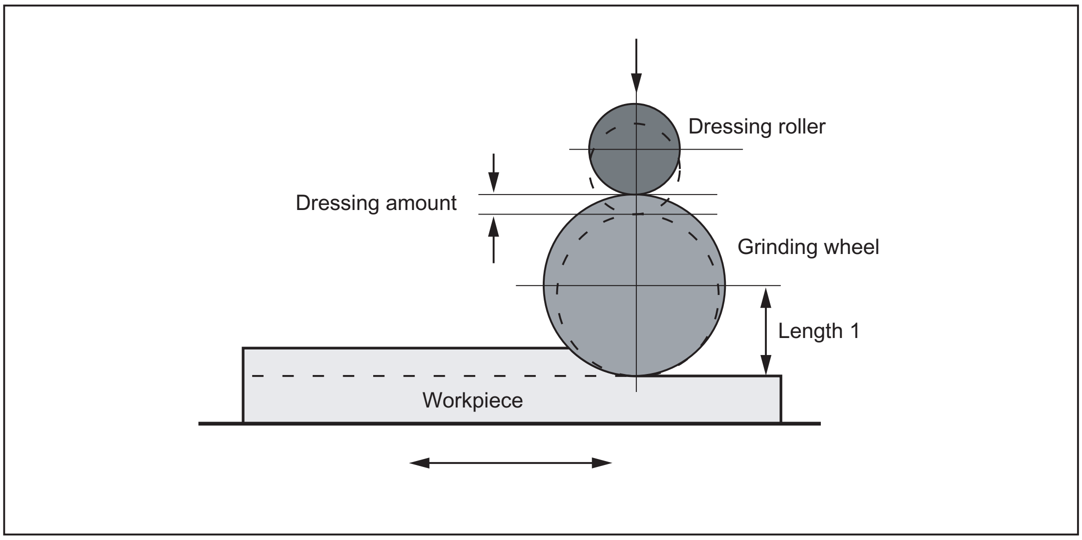


Figure 17-10 Dressing during machining using a dressing roller

General information

An online tool offset can be activated for every grinding tool in any channel.

The online tool offset is generally applied as a length compensation. Like geometry and wear data, lengths are assigned to geometry axes on the basis of the current plane as a function of the tool type.

The grinding spindle monitoring function remains active when an online tool offset is selected.

Note

The offset always corrects the wear parameters of the selected length. If the length compensation is identical for several cutting edges, then a chaining specification must be used to ensure that the values for the 2nd cutting edge are automatically corrected as well.

If online offsets are active in the machining channel, then the wear values for the active tool in this channel may not be changed from the machining program or via operator inputs.

Modifications to the radius wear (P15) are not taken into account until the tool is reselected.

The online offset is also applied to the constant grinding wheel peripheral speed (GWPS), i.e. the spindle speed is corrected by the corresponding value.

Instructions

The following commands are provided for online tool offsets:

Command	Meaning
`FCTDEF` <polynomial no.>, <lower limit>, <upper limit>, <coefficient 0>, <coefficient 1>, <coefficient 2>, <coefficient 3>)	Parameterize function (up to 3rd degree polynomial) (Fine Tool Offset Definition)
`PUTFTOCF` (<polynomial no.>, <reference value>, <length1_2_3>, <channel no.>, <spindle no.>)	Write online tool offset continuously (Put Fine Tool Offset Compensation)
`PUTFTOC` (<value>, <length1_2_3>, <channel no.>, <spindle no.>)	Write online tool offset discretely (Put Fine Tool Offset Compensation)
`FTOCON`	Activation of online tool offset (Fine Tool Offset Compensation ON)
`FTOCOF`	Deactivation of online tool offset (Fine Tool Offset Compensation OFF)

Note

Changes to the correction values in the TOA memory do not take effect until T or D is programmed again.

References:
Programming Manual, Job Planning

17.2.2 Write online tool offset: Continuous

FCTDEF

Certain dressing strategies (e.g. dressing roller) are characterized by the fact that the grinding wheel radius is continuously (linearly) reduced as the dressing roller is fed in. This strategy requires a linear function between infeed of the dressing roller and writing of the wear value of the respective length.

Function `FCTDEF` allows 3 independent functions to be defined according to the following syntax:

Function parameters

The function parameters are set in a separate block according to the following syntax:

FCTDEF(<polynomial no.>, <lower limit>, <upper limit>, <coefficient a0>, <coefficient a1>, <coefficient a2>, <coefficient a3>)

`FCTDEF`	Function definition
Polynomial no.:	Number of function (e.g. 1, 2 or 3)
Lower/upper limit:	Determines value range of the function (limit values in input resolutions)
Coefficients a_0, a_1, a_2:	Coefficients of polynomial

A 3rd degree polynomial is generally defined as follows:

$y = a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3$

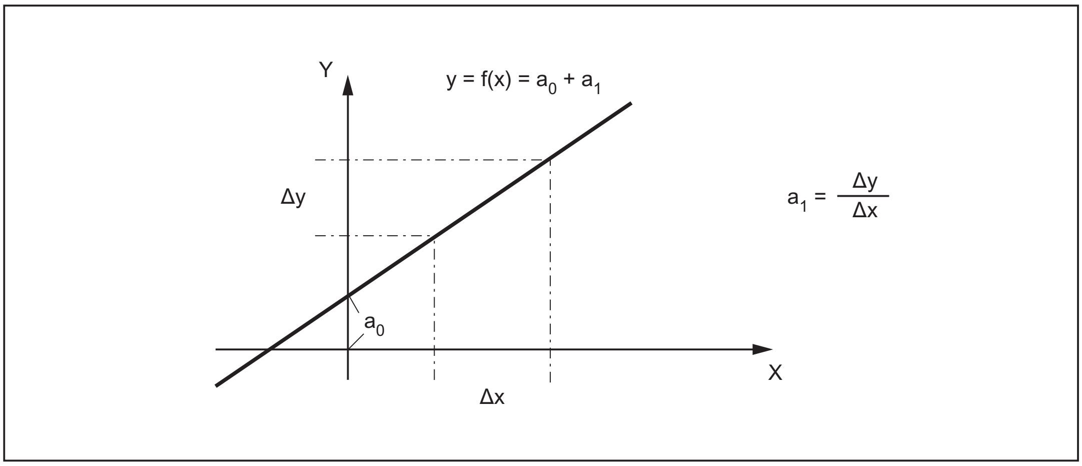


Figure 17-11 Straight line equation

Note

`FCTDEF` must be programmed in a separate NC block.

Example:

Existing conditions: Lead: $a_1 = +1$

$a_2 = 0$

$a_3 = 0$

At the time of definition, the function value y should be equal to 0 and should be derived from machine axis XA (e.g. dresser axis).

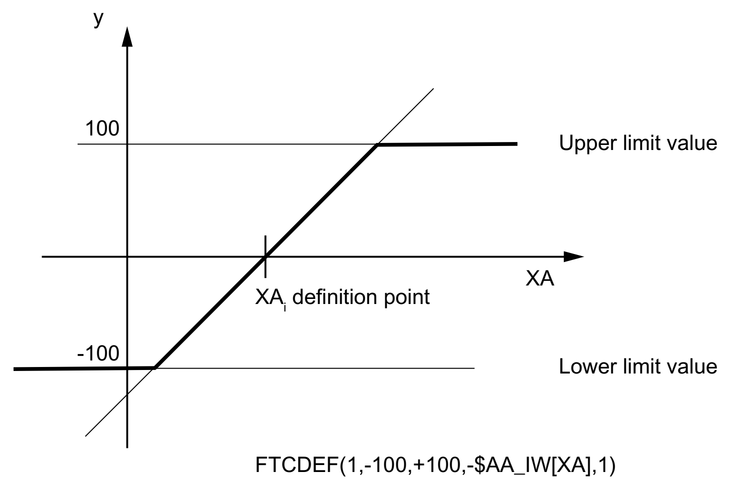


Figure 17-12 Straight line with gradient 1

Write online tool offset continuously

PUTFTOCF(<polynomial no.>, <reference value>, <length1_2_3>, <channel no.>, <spindle no.>)

PUFTOCF

Polynomial no.:	Number of function (1, 2, 3)
Reference value:	Reference value of function
Length 1_2_3:	Wear parameter into which the tool offset value is added
Channel no.:	Channel in which the offset is to be effective
Spindle no.:	Spindle for which the online offset is to be effective

The online tool offset is activated before the dresser axis movement block.

Example:

Program code	Comment
FCTDEF(1,-100,100,-$AA_IW[X],1)	; Function definition
PUTFTOCF(1,$AA_IW[X],1,2,1)	; Write online tool offset continuously

Length 1 of tool for spindle 1 in channel 2 is modified as a function of X axis movement.

Note

The online tool offset for a (geometric) grinding tool that is not active can be activated by specifying the appropriate spindle number.

If the channel number is omitted, the online offset is effective in the same channel.

If the spindle number is omitted, the online offset is applied to the current tool.

An online tool offset can also be called as a synchronized action.

References:
Function Manual, Synchronized Actions

17.2.3 Activate/deactivate online tool offset

Activation/deactivation of online tool offset

The following commands activate and deactivate the online tool offset in the machining channel (grinding, destination channel):

`FTOCON` Activation of online tool offset

The machining channel can process online tool offsets (`PUTFTOC`) only if the offset is active (`FTOCON`). Alarm 20204 "PUTFTOC command not allowed" is otherwise output.

`FTOCOF` Deactivation of online tool offset

`FTOCOF` deactivates the online tool offset. The written values remain stored in the appropriate length compensation data.

Online offsets are traversed in the basic coordinate system, i.e. even when the workpiece coordinate system has been rotated, the length compensations always act in parallel to the coordinates of the unrotated system.
The offset is applied regardless of whether or not the axis to be compensated is traversed in the current block.

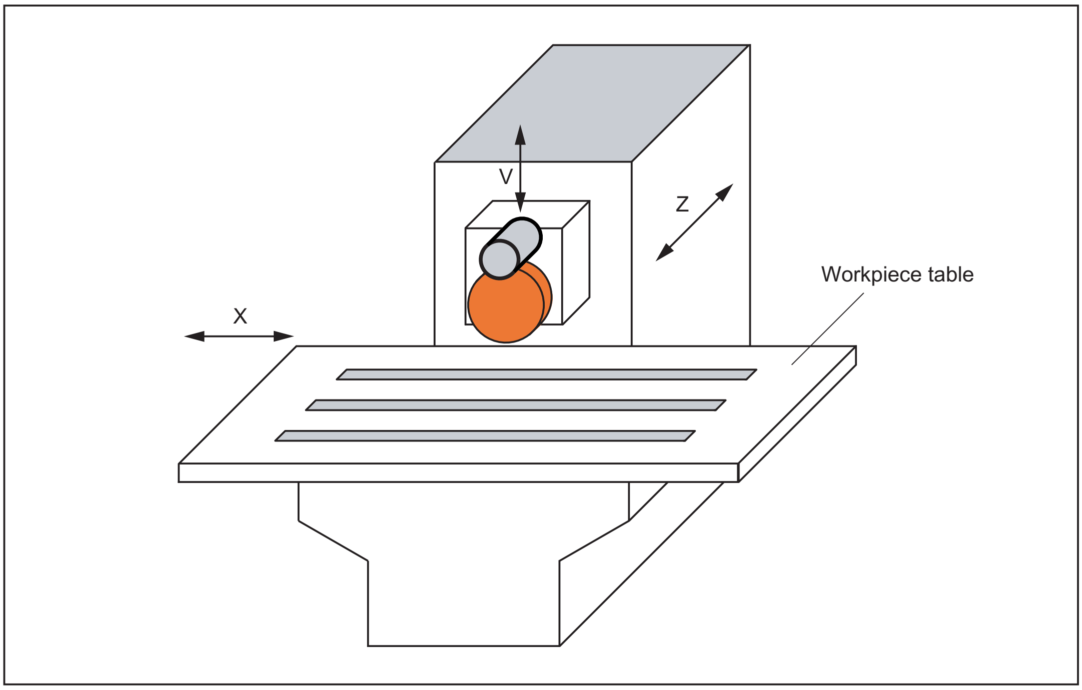
Note

Command `FTOCON` must be written to the channel in which the offset is to be applied (machining channel for grinding operation).

`FTOCOF` always corresponds to the reset position. `PUTFTOC` commands are effective only when the part program and `FTOCON` command are active.

17.2.4 Example of writing online tool offset continuously

Surface grinding machine



Y: Infeed axis for grinding wheel
V: Infeed axis for dressing roller
X: Reciprocating axis, left - right

Plane for the tool offset: `G19` (Y/Z plane)

Length 1 acts in Z, length 2 in Y, tool type = 401

Machining:	Channel 1 with axes Y, X
Dressing:	Channel 2 with axis V

Task

After the grinding operation has started at Y100, the grinding wheel must be dressed by 0.05 (in V direction). The dressing amount must be compensated continuously by means of an online offset.

Main machining program in channel 1

Program code	Comment
...	
G1 G19 F10 G90	; Basic position
T1 D1	; Select current tool
S100 M3 Y100	; Spindle On, traverse to
	; Starting position
FTOCON	; Activate online offset
INIT (2, "/_N_MPF_DIR/_N_ABRICHT_MPF", "S")	; Select program
	; in channel 2
START (2)	; Starting the program
	; in channel 2
Y200	; Travel to target position
...	
M30	

Dressing program in channel 2_N_ABRICHT_MPF

Program code	Comment
...	
FCTDEF (1, -1000, 1000, -$AA_IW[V], 1)	; Function definition
PUTFTOCF (1, $AA_IW[V], 2, 1)	; Write online tool offset
	; Continuous
U-0.05 G1 F0.01 G91	; Infeed movement to dress wheel
...	
M30	

Note

Axis V operates (dresses) in parallel to Y, i.e. length 2 acts in Y and must therefore be compensated.

17.2.5 Write online tool offset discretely

PUTFTOC

This command writes an offset value by means of a program command.

PUTFTOC(<value>, <length1_2_3>, <channel no.>, <spindle no.>)

Put Fine Tool Offset Compensation

The wear of the specified length (1, 2 or 3) is modified online by the programmed value.

Note

The online tool offset for a (geometric) grinding tool that is not active can be activated by specifying the appropriate spindle number.

If the channel number is omitted, the online offset is effective in the same channel.

If the spindle number is omitted, the online offset is applied to the current tool.

17.2.6 Information about online offsets

Response in the case of tool change

- In cases where `FTOCON` has been active since the last tool or cutting edge change, preprocessing stop with resynchronization is initiated in the control when a tool is changed.
- Cutting edge changes can be implemented without preprocessing stop.

Note

Tool changes can be executed in conjunction with the online tool offset through the selection of T numbers.

Tool changes with `M6` cannot be executed in conjunction with the online tool offset function.

Machining plane and transformation

- `FTOCON` can be used only in conjunction with the "Inclined axis" transformation.
- It is not possible to change transformations or planes (e.g. `G17` to `G18`) when `FTOCON` is active, except in the `FTOCOF` state.

Resets and operating mode changes

- When online offset is active, NC-STOP and program end with M2/M30 are delayed until the amount of compensation has been traversed.
- The online tool offset is immediately deselected in response to NC-RESET.
- Online tool offsets can be activated in AUTOMATIC mode and when the program is active.

Supplementary conditions

- The online tool offset is superimposed on the programmed axis motion, allowing for the defined limit values (e.g. velocity).

  If a DRF offset and online offset are active simultaneously for an axis, the DRF offset is considered first.
- The pending offset is traversed at JOG velocity, allowing for the maximum acceleration.

  In case of FTOCON the following channel-specific machine data is taken into account:

  MD20610 $MC_ADD_MOVE_ACCEL_RESERVE

  An acceleration margin can thus be reserved for the movement which means that the overlaid movement can be executed immediately.
- The valid online offset is deleted on reference point approach with G74.
- The fine offset is not deselected for tool changes with M6.

17.3 Online tool radius compensation

General information

When the longitudinal axis of the tool and the contour are perpendicular to each other, the offset can be applied as a length compensation to one of the three geometry axes (online tool length compensation).

If this condition is not fulfilled, then the offset quantity can be entered as a real radius compensation value (online tool radius compensation).

Enabling of function

The online tool-radius offset is activated via the machine data:

MD20254 $MC_ONLINE_CUTCOM_ENABLE (enable online tool radius compensation).

Activation/deactivation

An online tool radius compensation is activated and deactivated by means of commands FTOCON and FTOCOF
(in the same way as an online tool length compensation).

Parameterization

The parameters of the online tool offset are set using commands `PUTFTOCF` and `PUTFTOC`. Parameter "LENGTH 1_2_3" must be supplied as follows for an online tool radius compensation:

Parameter <length 1_2_3> = 4
Wear parameter to which correction value is added.

Supplementary conditions

- A tool radius compensation, and thus also an online tool radius compensation, can be activated only when the selected tool has a radius other than zero. This means that machining operations cannot be implemented solely with a tool radius compensation.
- The online offset values should be low in comparison to the original radius to prevent the permitted dynamic tolerance range from being exceeded when the offset is overlaid on the axis movement.
- When online tool radius compensation is applied to grinding and turning tools (types 400-599), the compensation value is applied as a function of the tool point direction, i.e. it acts as a radius compensation when tool radius compensation is active and as a length compensation when tool radius compensation is deactivated in the axes specified by the tool point direction.

  On all other tool types, the compensation value is applied only when tool radius compensation has been activated with `G41` or `G42`. The compensation value is canceled when tool radius compensation is deactivated with `G40`.

17.4 Grinding-specific tool monitoring

17.4.1 General information

Activation

The tool monitoring function is a combination of geometry and speed monitors and can be activated for any grinding tool (tool type: 400 to 499).

Selection

The monitoring function is selected:

- by programming (`TMON`) in the part program

  or

- automatically through selection of tool length compensation of a grinding tool with uneven tool type number.

Note

The automatic selection of the monitoring must be set via the channel-specific machine data:

MD20350 $MC_TOOL_GRIND_AUTO_TMON.

Monitoring active

The monitor for a grinding tool remains active until it is deselected again by means of program command `TMOF`.

Note

Monitoring of one tool is not deselected if the monitoring function is selected for another tool provided the two tools are referred to different spindles.

One tool and thus also **one** tool monitor can be active for every spindle at any point in time.

Activated monitors remain active after a RESET.

17.4.2 Geometry monitoring

Function

The following quantities can be monitored:

- The current grinding wheel radius

  and

- The current grinding wheel width

The current wheel radius is compared with the value stored in parameter $TC_TPG3.
The current radius is compared with the parameter number of the first edge (`D1`) of a grinding tool declared in parameter $TC_TPG9.

The current wheel width is generally calculated by the dressing cycle and can be entered in parameter $TC_TPG5 of a grinding tool. The value entered in this parameter is compared to the value stored in parameter $TC_TPG4 when the monitoring function is active.

When does monitoring take place?

The monitoring function for the grinding wheel radius remains active when an online tool offset is selected:

- When the monitoring function is activated
- when the current radius (online tool offset, wear parameter) or the current width ($TC_TPG5) is altered

Monitor reactions

If the current grinding wheel radius becomes smaller than the value stored in parameter $TC_TPG3 or the current grinding wheel width ($TC_TPG5) drops below the value defined in $TC_TPG4, the axis/spindle-specific bit DBX83.3 is set to "1" in DB31, ... at the PLC interface.
This bit is otherwise set to "0".

DB31, ... DBX83.3 = 1 ⇒ Geometry monitoring has responded

DB31, ... DBX83.3 = 0 ⇒ Geometry monitoring has not responded

---

Note

No error reaction is initiated internally in the control system.

17.4.3 Speed monitoring

Function

The speed monitor checks the grinding wheel peripheral speed (parameter $TC_TPG7) as well as the maximum spindle speed (parameter $TC_TPG6).

The unit of measurement is:

- Grinding wheel peripheral speed $m * s^{-1}$
- Spindle speed rev/min

Monitoring is cyclic. The value is always limited to the first limit value reached.

When does monitoring take place?

The speed setpoint is monitored against the speed limitation cyclically, allowing for the spindle override.

When is the speed limit value recalculated?

The speed limit value is recalculated:

- when the monitoring function is selected,
- when the online offset values (wear parameters) are altered.

Monitor reactions

The system reacts as follows when the speed monitor responds:

- The speed is restricted to the limit value

  and

- Interface signal:
  DB31, ... DBX83.6 (speed monitoring)
  is output.

DB31, ... DBX83.6 = 1 ⇒ Speed monitoring limit reached

DB31, ... DBX83.6 = 0 ⇒ Speed monitoring limit not reached

Note

No error reaction is initiated internally in the control system.

17.4.4 Selection/deselection of tool monitoring

Part program commands

The following part program commands are provided for selecting and deselecting the grinding-specific tool monitor of an active or inactive tool:

Command	Meaning
`TMON` Tool monitoring ON	Selection of tool monitoring for the active tool in the channel.
`TMOF` Tool monitoring OFF	Deselection of tool monitoring for the active tool in the channel.
`TMON` (T number) Tool monitoring ON (T No.)	Selection of tool monitoring for a non-active tool with T number.
`TMOF` (T number) Tool monitoring OFF (T No.)	Deselection of tool monitoring for a non-active tool with T number.
`TMOF` (0) Tool monitoring OFF (0)	Deselection of tool monitoring for all tools.

17.5 Constant grinding wheel peripheral speed (GWPS).

17.5.1 General information

What is GWPS?

A grinding wheel peripheral speed, as opposed to a spindle speed, is generally programmed for grinding wheels. This variable is determined by the technological process (e.g. grinding wheel characteristics, material pairing). The speed is then calculated from the programmed value and the current wheel radius.

Note

GWPS can be selected for grinding tools (types 400- 499).

Speed calculation

The formula for calculating the speed is as follows:

$$n\ [rpm] = \frac{GWP[m*s\text{-}1] * 60}{2\pi * R[m]}$$

Note

Grinding wheel peripheral speed can be programmed and selected for grinding tools (types 400- 499).

The wear is considered when calculating the radius (parameter $TC_TPG9).

This function also applies to inclined wheels/axes.

The associated wear and the base dimension as a function of the tool type are added to the parameter selected by $TC_TPG9.

The sum total is divided by "cos" ($TC_TPG8) when the value of parameter $TC_TPG8 (angle of inclined wheel) is positive, and is divided by "sin" ($TC_TPG8) when the value is negative.

When is the speed recalculated?

The speed is recalculated in response to the following events:

- GWPS programming
- Change in the online offset values (wear parameters).

17.5.2 Selection/deselection and programming of GWPS, system variable

Part program commands

The GWPS is selected and deselected with the following part program commands:

Command	Meaning
`GWPSON` Grinding wheel peripheral speed ON	Selection of GWPS for the active tool in the channel.
`GWPSOF` Grinding wheel peripheral speed OFF	Deselection of GWPS for the active tool in the channel.
`GWPSON`(T number) Grinding wheel peripheral speed ON (T no.)	Selection of GWPS for a non-active tool with T number.
`GWPSOF`(T number) Grinding wheel peripheral speed OFF (T no.)	Deselection of GWPS for a non-active tool with T number.
`S`[spindle number] = value	Programming of constant grinding wheel peripheral speed. Unit of value setting depends on basic system (m/s or ft/s).

References:
Programming Manual Fundamentals

Note

Parameter $TC_TPG1 assigns a spindle to the tool. Every following S value for this spindle is interpreted as a grinding wheel peripheral speed when GWPS is active (see above).

If GWPS is to be selected with a new tool for a spindle for which the GWPS function is already active, the active function must be deselected first with `GWPSOF` (otherwise an alarm is given out).

GWPS can be active simultaneously for several spindles, each with a different grinding tool, in the same channel.

Selection of GWPS with `GWPSON` does not automatically result in activation of tool length compensation or of the geometry and speed monitoring functions. When GWPS is deselected, the last speed to be calculated remains valid as the setpoint.

$P_GWPS[spindle number]

This system variable can be used to query from the sub-program whether the GWPS is active for a specific spindle.

TRUE : GWPS programming of spindle active
FALSE : GWPS programming of spindle not active

References:
Programming Manual Fundamentals

17.5.3 GWPS in all operating modes

General information

This function allows the constant grinding wheel peripheral speed (GWPS) function to be selected for a spindle immediately after POWER ON and to ensure that it remains active after an operating mode changeover, RESET or part program end.

The function is activated via the machine data:

MD35032 $MA_SPIND_FUNC_RESET_MODE (parameterization of the GWPS function)

GWPS after POWER ON

A grinding-specific tool is defined via the following machine data:

MD20110 $MC_RESET_MODE_MASK

MD20120 $MC_TOOL_RESET_VALUE

MD20130 $MC_CUTTING_EDGE_RESET_VALUE

Note

MD35032 $MA_SPIND_FUNC_RESET_MODE

If the above machine data is set and a grinding-specific tool (tool type 400 to 499, MD20110, MD20120, MD20130) is used with reference to a valid spindle (parameter $TC_TPG1), then GWPS is activated for that spindle.

GWPS is deselected for all other spindles in this channel.

GWPS after RESET/part program end

After a RESET/part program end, GWPS remains active for all spindles for which it was already selected.

Note

MD35032 $MA_SPIND_FUNC_RESET_MODE

If the machine data above is set and GWPS is active on RESET or part program end, then GWPS remains active for this spindle.

If machine data MD35032 $MA_SPIND_FUNC_RESET_MODE is **not** set and GWPS is active on RESET or part program end, then GWPS is deactivated for this spindle.

GWPS is deselected for all other spindles in this channel.

It can be determined as to whether the spindle continues to rotate with the actual speed after RESET using the following machine data:

MD35040 $MA_SPIND_ACTIVE_AFTER_RESET

Programming

The spindle speed can be modified through the input of a grinding wheel peripheral speed.

The spindle speed can be modified through:

- programming in the part program/overstoring
- programming the grinding wheel peripheral speed through assignment to address "S" in MDA
- spindle speed control via PLC (FC18).

DB31, ... DBX84.0 (GWPS active)

The following interface signal can be used to determine whether or not the GWPS is active:

DB31, ... DBX84.0 (GWPS active)

17.5.4 Programming example for GWPS

Data of tool T1 (peripheral grinding wheel)

```
$TC_DP1[1,1] = 403          ;Tool type
$TC_DP3[1,1] = 300          ;Length1
$TC_DP4[1,1] = 50           ;Length2
$TC_DP12[1,1] = 0           ;Wear length 1
$TC_DP13[1.1] =0            ;Wear length 2
$TC_DP21[1.1] =300          ;Base length 1
$TC_DP22[1.1] =400          ;Base length 2
$TC_TPG1[1] = 1             ;Spindle number
$TC_TPG8[1] = 0             ;Angle of inclined wheel
$TC_TPG9[1] = 3             ;Parameter no. for radius calculation
```

Data of tool T5 (inclined grinding wheel)

```
$TC_DP1[5,1] = 401          ;Tool type
$TC_DP3[5,1] = 120          ;Length1
$TC_DP4[5,1] = 30           ;Length2
$TC_DP12[5,1] = 0           ;Wear length 1
$TC_DP13[5,1] =0            ;Wear length 2
$TC_DP21[5,1] =100          ;Base length 1
$TC_DP22[5,1] =150          ;Base length 2
$TC_TPG1[5] = 2             ;Spindle number
$TC_TPG8[5] = 45            ;Angle of inclined wheel
$TC_TPG9[5] = 3             ;Parameter no. for radius calculation
```

Programming

Program code	Comment
N20 T1 D1	; Select T1 and D1
N25 S1=1000 M1=3	; 1000 rpm for spindle 1
N30 S2=1500 M2=3	; 1500 rpm for spindle 2
...	
N40 GWPSON	; ;Selection of GWPS for active tool T1
N45 S[$P_AGT[1]]=60	; Set GWPS to 60 m/s for active tool n=1909.85 rpm
...	
N50 GWPSON(5)	; GWPS selection for tool 5 (2nd spindle)
N55 S[$TC_TPG1[5]]=40	; Set GWPS to 40 m/s for spindle 2 n=1909.85 rpm
...	
N60 GWPSOF	; Deactivate GWPS for active tool
N65 GWPSOF(5)	; Switch off GWPS for tool 5 (spindle 2)
...	

For more information, see Section "P5: Oscillation - only 840D sl (Page 641)".

Supplementary references

- Function Manual, Basic Functions; Feedrates (V1)
- Function Manual, Synchronized Actions

17.6 Supplementary Conditions

17.6.1 Tool changes with online tool offset

Tool change

Tool changes with `M6` cannot be executed in conjunction with the online tool offset function.

17.7 Data lists

17.7.1 Machine data

17.7.1.1 General machine data

Number	Identifier: $MN_	Description
18094	MM_NUM_CC_TDA_PARAM	Number of TDA
18096	MM_NUM_CC_TOA_PARAM	Number of TOA
18100	MM_NUM_CUTTING_EDGES_IN_TOA	Tool offsets per TOA

17.7.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20254	ONLINE_CUTCOM_ENABLE	Enable online tool radius compensation
20350	TOOL_GRIND_AUTO_TMON	Automatic tool monitoring
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements

17.7.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32020	JOG_VELO	JOG axis velocity
35032	SPIND_FUNC_RESET_MODE	Parameterization of GWPS function

17.7.2 Signals

17.7.2.1 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Geometry monitoring	DB31, ...DBX83.3	DB390x.DBX2001.3
Speed monitoring	DB31, ...DBX83.6	DB390x.DBX2001.6
GWPS active	DB31, ...DBX84.1	DB390x.DBX2002.1

18 Z2: NC/PLC interface signals

18.1 Digital and analog NCK I/Os (A4)

18.1.1 Signals to NC (DB10)

Overview of signals from PLC to NC

DB10	Signals to NC interface PLC → NC							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Disable digital NCK inputs							
	Digital inputs without hardware *)				(on-board inputs) **)			
	Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
1	Setting by PLC of the digital NCK inputs							
	Digital inputs without hardware *)				(on-board inputs) **)			
	Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
4	Disable digital NCK outputs							
	Digital outputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
5	Overwrite screen form for digital NCK outputs							
	Digital outputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
6	Set value by PLC of the digital NCK outputs							
	Digital outputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
7	Setting screen form for digital NCK outputs							
	Digital outputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
Notes: *) Bits 4 to 7 of the digital NCK inputs/outputs can be processed by the PLC even though there are no equivalent hardware I/Os. These bits can therefore also be used for data exchange between the NCK and PLC. **) The NCK digital inputs/outputs 1 to 4 are provided as onboard hardware inputs and outputs.								
DB10	Signals to NC interface PLC → NC							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
122	Disable digital NCK inputs							
	Input 16	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9
123	Setting by PLC of the digital NCK inputs							
	Input 16	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9

124	Disable digital NCK inputs							
	Input 24	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17
125	Setting by PLC of the digital NCK inputs							
	Input 24	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17
126	Disable digital NCK inputs							
	Input 32	Input 31	Input 30	Input 29	Input 28	Input 27	Input 26	Input 25
127	Setting by PLC of the digital NCK inputs							
	Input 32	Input 31	Input 30	Input 29	Input 28	Input 27	Input 26	Input 25
128	Disable digital NCK inputs							
	Input 40	Input 39	Input 38	Input 37	Input 36	Input 35	Input 34	Input 33
129	Setting by PLC of the digital NCK inputs							
	Input 40	Input 39	Input 38	Input 37	Input 36	Input 35	Input 34	Input 33
130	Disable digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
131	Overwrite screen form for digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
132	Set value by PLC of the digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
133	Setting screen form for digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
134	Disable digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
135	Overwrite screen form for digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
136	Set value by PLC of the digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
137	Setting screen form for digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
138	Disable digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
139	Overwrite screen form for digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
140	Set value by PLC of the digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
141	Setting screen form for digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
142	Disable digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
143	Overwrite screen form for digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
144	Set value by PLC of the digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
145	Setting screen form for digital NCK outputs							

	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
146	Disable analog NCK inputs							
	Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
147	Setting screen form for analog NCK inputs							
	Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
148, 149	Setting value from PLC for analog input 1 of the NCK							
150, 151	Setting value from PLC for analog input 2 of the NCK							
152, 153	Setting value from PLC for analog input 3 of the NCK							
154, 155	Setting value from PLC for analog input 4 of the NCK							
156, 157	Setting value from PLC for analog input 5 of the NCK							
158, 159	Setting value from PLC for analog input 6 of the NCK							
160, 161	Setting value from PLC for analog input 7 of the NCK							
162, 163	Setting value from PLC for analog input 8 of the NCK							
166	Overwrite screen form for analog NCK outputs							
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
167	Setting screen form for analog NCK outputs							
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
168	Disable analog NCK outputs							
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
170, 171	Setting value from PLC for analog output 1 of NCK							
172, 173	Setting value from PLC for analog output 2 of NCK							
174, 175	Setting value from PLC for analog output 3 of NCK							
176, 177	Setting value from PLC for analog output 4 of NCK							
178, 179	Setting value from PLC for analog output 5 of NCK							
180, 181	Setting value from PLC for analog output 6 of NCK							
182, 183	Setting value from PLC for analog output 7 of NCK							
184, 185	Setting value from PLC for analog output 8 of NCK							

Description of signals from PLC to NC

DB10 DBB0, 122, 124, 126, 128	Disable digital NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The digital input of the NCK is disabled by the PLC. It is thus set to "0" in a defined way in the control.
Signal state 0 or edge change 1 → 0	The digital input of the NCK is enabled. The signal state applied at the input can now be read directly in the NC part program.
Corresponding to	DB10 DBB1 (Setting by PLC of the digital NCK inputs) DB10 DBB60 (actual value for digital NCK inputs) MD10350 $MN_FASTIO_DIG_NUM_INPUTS

DB10 DBB1, 123, 125, 127, 129	Setting by PLC of the digital NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The digital NCK input is set to a defined "1" state by the PLC. This means the signal state at the hardware input and disabling of the input (IS "Disable the digital NCK inputs") have no effect.
Signal state 0 or edge change 1 → 0	The signal state at the NCK input is enabled for read access by the NC part program. However, the state can be accessed only if the NCK input is not disabled by the PLC (IS "Disable digital NCK inputs" = 0).
Corresponding to ...	DB10 DBB0 (Disable digital NCK inputs) DB10 DBB60 (actual value for digital NCK inputs) MD10350 $MN_FASTIO_DIG_NUM_INPUTS

DB10 DBB4, 130, 134, 138, 142	Disable digital NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The digital NCK output is disabled. "0V" is output in a defined way at the hardware output.
Signal state 0 or edge change 1 → 0	The digital output of the NCK is enabled. As a result, the value set by the NC part program or the PLC is output at the hardware output.
Corresponding to ...	DB10 DBB5 (Overwrite screen form for digital NCK outputs) DB10 DBB7 (Setting screen form for digital NCK outputs) DB10 DBB6 (Setting by PLC of digital NCK outputs) MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB5, 131, 135, 139, 143	Overwrite screen form for digital NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	On signal transition 0 → 1 the previous NCK value is overwritten by the setting value (IS "Set value by PLC of the digital NCK outputs"). The previous NCK value, which, for example, was directly set by the part program, is lost. The signal status defined by the setting value forms the new NCK value.
Signal state 0 or edge change 1 → 0	As the interface signal is only evaluated by the NCK on signal transition 0 →1 it must be reset to "0" again by the PLC user program in the next PLC cycle.
Special cases, errors, ...	**Note:** The PLC interface for the setting value (DB10, DBB6) is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms via the PLC user program must be avoided.
Corresponding to	DB10 DBB4 (Disable digital NCK outputs) DB10 DBB7 (Setting screen form for digital NCK outputs) DB10 DBB6 (Setting value by PLC of digital NCK outputs) MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB6, 132, 136, 140, 144	**Set value by PLC of the digital NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The signal status for the digital hardware output can be changed by the PLC with the setting value. There are two possibilities: • With the "Overwrite screen form": With signal transition 0 → 1 in the 'overwrite screen form' the PLC overwrites the previous 'NCK value' with the 'setting value'. This is the new 'NCK value'. • With the 'setting screen form': On signal state 1 in the "setting screen form", the "PLC value" is activated. The value used is the 'setting value'. With "setting value" "1", signal level 1 is put out at the hardware output. With "0", the output level is 0. The associated voltage values can be found in: **References:** Device Manual, NCU 7x0.3 PN
Signal state 0 or edge change 1 → 0	As the interface signal is only evaluated by the NCK on signal transition 0 → 1 it must be reset to "0" again by the PLC user program in the next PLC cycle.
Special cases, errors, ...	**Note:** The PLC interface for the setting value (DB10, DBB6) is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms via the PLC user program must be avoided.
Corresponding to	DB10 DBB4 (Disable digital NCK outputs) DB10 DBB5 (Overwrite screen form for digital NCK outputs) DB10 DBB7 (Setting screen form for digital NCK outputs) MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB7, 133, 137, 141, 145	**Setting screen form for digital NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Instead of the NCK value, the PLC value is output at the digital hardware output. The PLC value must first be deposited in IS "Set value by PLC of the digital NCK outputs". The current NCK value is not lost.
Signal state 0 or edge change 1 → 0	The NCK value is output at the digital hardware output.
Special cases, errors, ...	**Note:** The PLC interface for the setting value (DB10, DBB6) is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms via the PLC user program must be avoided.
Corresponding to	DB10 DBB4 (Disable digital NCK outputs) DB10 DBB5 (Overwrite screen form for digital NCK outputs) DB10 DBB6 (Setting value by PLC of digital NCK outputs) MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB146	Disable analog NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The analog input of the NCK is disabled by the PLC. It is thus set to "0" in a defined way in the control.
Signal state 0 or edge change 1 → 0	The analog input of the NCK is enabled. This means that the analog value at the input can be read directly in the NC part program if the setting screen form is set to 0 signal by the PLC for this NCK input.
Corresponding to ...	DB10 DBB147 (Setting screen form of analog NCK inputs) DB10 DBB148 (Setting by PLC of analog NCK inputs) DB10 DBB199 ... (Actual value of analog NCK inputs) MD10300 $MN_FASTIO_ANA_NUM_INPUTS

DB10 DBB147	Setting screen form of analog NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The setting value from the PLC acts as the enabled analog value (IS "Setting value from PLC for analog NCK inputs").
Signal state 0 or edge change 1 → 0	The analog value at the NCK input is enabled for read access by the NC part program However, the state can be accessed only if the NCK input is not disabled by the PLC (IS "Disable analog NCK inputs" = 0).
Corresponding to	DB10 DBB146 (Disable analog NCK inputs) DB10 DBB148 to 163 (Setting by PLC of analog NCK inputs) DB10 DBB199-209 (Actual value of analog NCK inputs) MD10300 $MN_FASTIO_ANA_NUM_INPUTS

DB10 DBB148 - 163	Setting value from PLC for analog NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	With this setting value a defined analog value can be set by the PLC. With IS "Setting screen form of analog NCK inputs", the PLC selects whether the analog value at the hardware input or the setting value from the PLC is to be used as the enabled analog value. The setting value from the PLC becomes active as soon as IS "Setting screen form" is set to "1". The setting value from the PLC is specified as a fixed point number (16 bit value including sign) in 2's complement.
Corresponding to	DB10 DBB146 (Disable analog NCK inputs) DB10 DBB147 (Setting screen form of analog NCK inputs) DB10 DBB199-209 (Actual value of analog NCK inputs) MD10300 $MN_FASTIO_ANA_NUM_INPUTS

DB10 DBB166	**Overwrite screen form for analog NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	On signal transition 0 → 1 the previous NCK value is overwritten by the setting value (IS "Setting value from PLC for analog NCK outputs"). The previous NCK value which, for example, was directly set by the part program, is lost. The analog value specified by the PLC setting value forms the new NCK value.
Signal state 0 or edge change 1 → 0	As the interface signal is only evaluated by the NCK on signal transition 0 → 1 it must be reset to "0" again by the PLC user program in the next PLC cycle.
Special cases, errors, ...	**Note:** The PLC interface for the setting value is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program.
Corresponding to	DB10 DBB168 (Disable analog NCK outputs) DB10 DBB167 (Setting screen form of analog NCK outputs) DB10 DBB170-185 (Setting by PLC of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

DB10 DBB167	**Setting screen form of analog NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Instead of the NCK value, the PLC value is output at the analog hardware output. The PLC value must first be stored in IS "Setting value from PLC for the analog NCK outputs". The current NCK value is not lost.
Signal state 0 or edge change 1 → 0	The NCK value is output at the analog hardware output.
Special cases, errors, ...	**Note:** The PLC interface for the setting value is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program.
Corresponding to	DB10 DBB168 (Disable analog NCK outputs) DB10 DBB166 (Overwrite screen form of analog NCK outputs) DB10 DBB170-185 (Setting by PLC of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

DB10 DBB168	**Disable analog NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The analog output of the NCK is disabled. "0V" is output in a defined way at the hardware output.
Signal state 0 or edge change 1 → 0	The analog output of the NCK is enabled. As a result, the value set by the NC part program or the PLC is output at the hardware output.
Corresponding to	DB10 DBB166 (Overwrite screen form of analog NCK outputs) DB10 DBB167 (Setting screen form of analog NCK outputs) DB10 DBB170-185 (Setting by PLC of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

DB10 DBB170 - 185	**Setting value from PLC for analog NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	With this setting value, the value for the analog hardware output can be changed by the PLC. There are two possibilities: • With the "Overwrite screen form": With signal transition 0 → 1 in the 'overwrite screen form' the PLC overwrites the previous 'NCK value' with the 'setting value'. This is the new "NCK value". • With the "setting screen form": On signal state 1 in the "setting screen form", the "PLC value" is activated. The value used is the 'setting value'. The setting value from the PLC is specified as a fixed point number (16 bit value including sign) in 2's complement.
Signal state 0 or edge change 1 → 0	As the interface signal is only evaluated by the NCK on signal transition 0 → 1 it must be reset to "0" again by the PLC user program in the next PLC cycle.
Special cases, errors, ...	**Note:** The PLC interface for the setting value is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program.
Corresponding to	DB10 DBB168 (Disable analog NCK outputs) DB10 DBB166 (Overwrite screen form of analog NCK outputs) DB10 DBB167 (Setting screen form of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

18.1.2 Signals from NC (DB10)

Overview of signals from NC to PLC

DB10	Signals from NC interface NC → PLC							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
60	Actual value for digital NCK inputs							
					(on-board inputs) **)			
					Input 4	Input 3	Input 2	Input 1
64	Setpoint for digital NCK outputs							
	Digital inputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
186	Actual value for digital NCK inputs							
	Input 16	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9
187	Actual value for digital NCK inputs							
	Input 24	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17
188	Actual value for digital NCK inputs							
	Input 32	Input 31	Input 30	Input 29	Input 28	Input 27	Input 26	Input 25
189	Actual value for digital NCK inputs							
	Input 40	Input 39	Input 38	Input 37	Input 36	Input 35	Input 34	Input 33
190	Setpoint for digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
191	Setpoint for digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
192	Setpoint for digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
193	Setpoint for digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
Notes: *) Bits 4 to 7 of the digital inputs and NCK outputs can be processed by the PLC although no equivalent hardware I/Os exist. These bits can therefore also be used for data exchange between the NCK and PLC. **) The NCK digital inputs/outputs 1 to 4 are provided as onboard hardware inputs and outputs.								
DB10	Signals from NC interface NC → PLC							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
194, 195	Actual value for analog input 1 of NCK							
196, 197	Actual value for analog input 2 of NCK							
198, 199	Actual value for analog input 3 of NCK							

200, 201	Actual value for analog input 4 of NCK
202, 203	Actual value for analog input 5 of NCK
204, 205	Actual value for analog input 6 of NCK
206, 207	Actual value for analog input 7 of NCK
208, 209	Actual value for analog input 8 of NCK
210, 211	Setpoint for analog output 1 of NCK
212, 213	Setpoint for analog output 2 of NCK
214, 215	Setpoint for analog output 3 of NCK
216, 217	Setpoint for analog output 4 of NCK
218, 219	Setpoint for analog output 5 of NCK
220, 221	Setpoint for analog output 6 of NCK
222, 223	Setpoint for analog output 7 of NCK
224, 225	Setpoint for analog output 8 of NCK

Description of signals from NC to PLC

DB10 DBB60, 186 - 189	Actual value for digital NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Signal level "1" is active at the digital hardware input of the NCK.
Signal state 0 or edge change 1 → 0	Signal level "0" is active at the digital hardware input of the NCK.
Special cases, errors, ...	The influence of interface signal: DB10 DBB0 (Disable digital NCK inputs) is ignored for the actual value.
Corresponding to	DB10 DBB0 (Disable digital NCK inputs) MD10350 $MN_FASTIO_DIG_NUM_INPUTS

DB10 DBB64, 190 - 193	**Setpoint for digital NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The NCK value for the digital output currently set (setpoint) is "1".
Signal state 0 or edge change 1 → 0	The NCK value for the digital output currently set (setpoint) is "0".
Signal irrelevant for ...	This 'setpoint' is only output to the hardware output under the following conditions: • Output is not disabled (IS "Disable digital NCK outputs") • PLC has switched to the NCK value (IS "Setting screen form for digital NCK inputs") As soon as these conditions are fulfilled, the "setpoint" of the digital output corresponds to the "actual value".
Corresponding to	DB10 DBB4 (Disable digital NCK outputs) DB10 DBB5 (Overwrite screen form for digital NCK outputs) DB10 DBB6 (Setting value by PLC of digital NCK outputs) DB10 DBB7 (Setting mask for digital NCK outputs) MD10310 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB194 - 209	**Actual value for analog NCK inputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The analog value applied to the analog NCK input is signalled to the PLC. The actual value is set as a fixed point number (16 bit value including sign) in 2's complement by the NCK.
Signal state 0 or edge change 1 → 0	The effect of the PLC on the analog value (e.g. with IS "Disable analog NCK inputs") is ignored.
Corresponding to	DB10 DBB146 (Disable analog NCK inputs) DB10 DBB147 (Setting screen form of analog NCK inputs) DB10 DBB148-163 (Setting by PLC of analog NCK inputs) MD10300 $MN_FASTIO_ANA_NUM_INPUTS

DB10 DBB210 - 225	**Setpoint for analog NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The current set NCK value for the analog output (setpoint) is signalled to the PLC. The set value is set as a fixed point number (16 bit value including sign) in 2's complement by the NCK.

DB10 DBB210 - 225	Setpoint for analog NCK outputs
Signal state 0 or edge change 1 → 0	This 'setpoint' is only output to the hardware output under the following conditions: • Output is not disabled (IS "Disable analog NCK outputs") • The PLC has switched to the NCK value (IS "Setting screen form of analog NCK outputs")
Corresponding to	DB10 DBB168 (Disable analog NCK outputs) DB10 DBB166 (Overwrite screen form of analog NCK outputs) DB10 DBB170-185 (Setting by PLC of analog NCK outputs) DB10 DBB167 (Setting screen form of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

18.2 Distributed systems (B3)

18.2.1 Defined logical functions/defines

BUSTYP

Name	Value	Interface DB19	Meaning
`MPI`	1	DBW100.102.104. 120. 130 Bits 8 -15	Control unit to MPI, 187.5 kbaud
`OPI`	2	"	Control unit to MPI, 1.5 Mbaud

STATUS

Name	Value	Interface DB19	Meaning
`OFFL_REQ_PLC`	1	Online interfaces 1. : DBB124 2. : DBB134	PLC to control unit: PLC wants to displace control unit by offline request.
`OFFL_CONF_PLC`	2	Online interfaces 1. : DBB124 2. : DBB134	Control unit to PLC: Acknowledgement of `OFFL_REQ_PLC` The meaning of the signal is dependent on Z_INFO DBB125 or DBB135
`OFFL_REQ_OP`	3	Online interfaces 1. : DBB124 2. : DBB134	Control unit to PLC: Control unit would like to go offline from this NCU and outputs an offline request
`OFFL_CONF_OP`	4	Online interfaces 1. : DBB124 2. : DBB134	PLC to control unit: Acknowledgement of `OFFL_REQ_OP` The meaning of the signal is dependent on Z_INFO DBB125 or DBB135

Name	Value	Interface DB19	Meaning
ONL_PERM	5	Online request interface DBB108	PLC to control unit: PLC notifies control unit as to whether it can go online or not. The meaning of the signal is dependent on Z_INFO: DBB109
S_ACT	6	Online interfaces 1. : DBB124 2. : DBB134	Control unit to PLC: Control unit goes online or changes operating focus. The meaning of the signal is dependent on Z_INFO DBB125 or DBB135
OFFL_REQ_FOC	7	Online interfaces 1. : DBB124 2. : DBB134	Control unit to PLC: Control unit would like to take operating focus away from this NCU
OFFL_CONF_FOC	8	Online interfaces 1. : DBB124 2. : DBB134	PLC to control unit: Acknowledgement of OFFL_REQ_FOC The meaning of the signal is dependent on Z_INFO DBB125 or DBB135
ONL_REQ_FOC	9	Online interfaces 1. : DBB124 2. : DBB134	Control unit to PLC: Control unit would like to set operating focus to this NCU
ONL_PERM_FOC	10	Online interfaces 1. : DBB124 2. : DBB134	PLC to control unit: Acknowledgement of ONL_REQ_FOC The meaning of the signal is dependent on Z_INFO DBB125 or DBB135

Z_INFO

Name	Value	Interface DB19	Meaning
DISC_FOC	9	DBB125 DBB135	Control unit switches operating focus to another NCU.
Set	10	DBB109 Bit 0-3 DBB125 DBB135	Positive acknowledgement
CONNECT	11	DBB125 DBB135	Control unit has gone online on this NCU.
MMC_LOCKED	13	DBB109 Bit 0-3 DBB125 DBB135	HMI has set switchover disable. There are processes running on this control unit that may not be interrupted by a switchover.
PLC_LOCKED	14	DBB109 Bit 0-3 DBB125 DBB135	The HMI switchover disable is set in the HMI-PLC interface. Control unit cannot go offline from this NCU or change operating focus.
PRIO_H	15	DBB109 Bit 0-3 DBB125 DBB135	Control units with a higher priority are operating on this NCU. Requesting control unit cannot go online to this NCU.

STATUS and Z_INFO can be combined as follows

Name: Status	Z_INFO	Meaning
OFFL_REQ_PLC	Set	PLC wants to displace online control unit by offline request.
OFFL_CONF_PLC	Set	Control unit positively acknowledges the offline request from PLC. Control unit will subsequently go offline.
OFFL_CONF_PLC	MMC_LOCKED	Control unit negatively acknowledges the offline request. Control unit will not go offline, as processes are running that must not be interrupted.
OFFL_REQ_OP	Set	Control unit would like to go offline from the online NCU and outputs an offline request.
OFFL_CONF_OP	Set	PLC positively acknowledges the offline request. Control unit will subsequently go offline from this NCU.
OFFL_CONF_OP	PLC_LOCKED	PLC negatively acknowledges the offline request from control unit. User has set the HMI switchover disable, control unit cannot go offline, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, 1st or 2nd HMI-PLC interface.
ONL_PERM	No. of HMI-PLC online interface, OK	PLC issues the online enabling command to the requesting control unit. Control unit can then go online to this NCU. Content of Z_INFO: Bit 0 ..3: Set Bit 4... 7: No. of HMI-PLC online interface with which the control unit should connect: First HMI-PLC online interface Second HMI-PLC online interface
ONL_PERM	MMC_LOCKED	The requesting control unit cannot go online. Two control units on which uninterruptible processes are in progress are connected online to this NCU. The PLC cannot suppress either of the two control units.
ONL_PERM	PLC_LOCKED	The requesting control unit cannot go online. User has set HMI switchover disable, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, first or second HMI online interface.
ONL_PERM	PRIO_H	The requesting control unit cannot go online. Two control units that are both higher priority than the requesting control unit are connected online to the NCU. The PLC cannot suppress either of the two control units.
S_ACT	CONNECT	The requesting control unit has gone online. The PLC now activates HMI sign-of-life monitoring.
S_ACT	DISC_FOCUS	Server HMI has disconnected the operating focus from this NCU.
OFFL_REQ_FOC	Set	Server HMI would like to disconnect the operating focus from this NCU and outputs an offline focus request.
OFFL_CONF_FOC	Set	PLC positively acknowledges the offline focus request. Server HMI can disconnect operating focus.
OFFL_CONF_FOC	PLC_LOCKED	PLC negatively acknowledges the online focus request. User has set HMI switchover disable, server HMI cannot disconnect operating focus, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, first or second HMI-PLC interface.

Name: Status	Z_INFO	Meaning
ONL_REQ_FOC	Set	Server HMI would like to set the operating focus on this NCU and outputs an online focus request.
ONL_PERM_FOC	Set	PLC positively acknowledges the online focus request. Server HMI then connects operating focus to this NCU.
ONL_PERM_FOC	PLC_LOCKED	PLC negatively acknowledges the online focus request. User has set HMI switchover disable, server HMI cannot set operating focus, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, first or second HMI-PLC interface.

18.2.2 Interfaces in DB19 for M:N

The HMI/PLC interface in DB19 is divided into three areas.

Online request interface

The online request sequence is executed on this interface if a control unit wants to go online.

HMI writes its client ID to ONL_REQUEST and waits for the return of the client ID in

ONL_CONFIRM.

After the positive acknowledgement from the PLC, the control unit sends its parameters and waits for online permission (in PAR_STATUS, PAR_Z_INFO).

HMI parameter transfer:

Client identification -> PAR_CLIENT_IDENT

HMI type -> PAR_MMC_TYP

MCP address -> PAR_MSTT_ADR

With the positive online permission, the PLC also sends the number of the HMI-PLC online interface DBB109.4-7 to be used by the control unit.

The MMC then goes online and occupies the online interface assigned by the PLC.

Online interfaces

Two control units can be connected online to one NCU at the same time.

The online interface is available for each of the two online control units separately.

After a successful online request sequence, the control unit receives the number of its online interface from the PLC.

The HMI parameters are then transferred to the corresponding online interface by the PLC.

The control unit goes online and occupies its own online interface via which data are then exchanged between the HMI and PLC.

HMI data interfaces

User data from/to the HMI are defined on these:

- DBB 0-49 control unit 1 interface
- DBB 50-99 control unit 2 interface

These data and signals are always needed to operate control units.

M:N sign-of-life monitoring

This is an additional monitoring function which must not be confused with the HMI sign-of-life monitor. For further information, please refer to the relevant signals.

In certain operating states, control units with activated M:N switchover (parameterizable in NETNAMES.INI) must be capable of determining from a PLC data whether they need to wait or not before linking up with an NCU.

Example:

Control units with an activated control unit switchover function must be capable of starting up an NCU without issuing an online request first.

Control unit must go online for service-related reasons.

The operation is coordinated in the online request interface via data DBW110:
M_TO_N_ALIVE

The M:N sign of life is a ring counter which is incremented cyclically by the PLC or set to a value of 1 when it overflows.

Before a control unit issues an online request, it must check the sign of life to establish whether the M:N switchover is activated in the PLC.

Procedure:

HMI reads the sign of life at instants T0 and T0 + 1.

Case 1: negative acknowledgement for reading process, DB19 does not exist. Control unit goes online without request procedure.

Case 2: m_to_n_alive = 0, control unit switchover disabled. Control unit goes online without request procedure.

Case 3: m_to_n_alive (T0) = m_to_n_alive (T0+1), control unit switchover disabled. Control unit goes online without request procedure.

Case 4: m_to_n_alive (T0) <> m_to_n_alive (T0+1), control unit switchover enabled.

Case 1 ... case 3 apply only under special conditions and not in normal operation.

Online request interface

DB19 DBW100	ONL_REQUEST
Client_Ident	Control unit would like to go online and use the online request interface. HMI first writes its Client_Ident as a request. Bit 8 .. 15: Bus type: MPI 1 or BTSS 2 Bit 0 .. 7: HMI bus address
DB19 DBW102	ONL_CONFIRM.
Client_Ident	If the online request interface is not being used by another control unit, the PLC returns the Client identification as positive acknowledgement. Bit 8 .. 15: Bus type: MPI 1 or BTSS 2 Bit 0 .. 7: HMI bus address
DB19 DBW104	PAR_CLIENT_IDENT HMI parameter transfer to PLC
Client_Ident	Bit 8 .. 15: Bus type: MPI 1 or BTSS 2 Bit 0 .. 7: HMI bus address
DB19 DBB106	PAR_MMC_TYP HMI parameter transfer to PLC
HMI type from NETNAMES.INI	Type properties of the control unit configured in file NETNAMES.INI. Evaluated by the PLC when MMC is suppressed (server, main/secondary operator panel, ...), see description of file NETNAMES.INI
DB19 DBB107	PAR_MSTT_ADR HMI parameter transfer to PLC
MCP address from NETNAMES.INI	Address of MCP to be switched over or activated/deactivated with the control unit. Parameter from NETNAMES.INI
255	No MCP is assigned to control unit, no MCP will be activated/deactivated
DB19 DB108	PAR_STATUS PLC sends HMI pos./neg. online permission
ONL_PERM (5)	PLC notifies HMI as to whether control unit can go online or not. The meaning of the signal is dependent on PAR_Z_INFO:
DB19 DBB109	PAR_Z_INFO PLC sends HMI pos./neg. online permission
No. of HMI-PLC online interface, OK (10)	PLC issues the online enabling command to the requesting control unit. Control unit can then go online to this NCU. Bit 0 ..3: Set Bit 4 .. 7: No. of HMI-PLC online interface with which the control unit should connect: First HMI-PLC online interface Second HMI-PLC online interface
MMC_LOCKED (13)	The requesting control unit cannot go online. Two control units on which uninterruptible processes are in progress are connected online to this NCU. The PLC cannot suppress either of the two control units.
PLC_LOCKED (14)	The control unit switchover disable is set in the HMI-PLC interface.
PRIO_H (15)	The requesting control unit cannot go online. Two control units that are both higher priority than the requesting control unit are connected online to the NCU. The PLC cannot suppress either of the two control units.

Sign of life of M:N switchover

DB19 DBW110	M_TO_N_ALIVE
1 ... 65535	Ring counter that is cyclically incremented by the PLC. Indicator for the HMI that the M:N switchover is active and ready.

1. HMI/PLC online interface

DB19 DBW120	MMC1_CLIENT_IDENT
	refer to PAR_CLIENT_IDENT after issuing positive online permission, the PLC transfers the HMI parameters to the online interface PAR_CLIENT_IDENT -> MMC1_CLIENT_IDENT
DB19 DBB122	MMC1_TYP
	refer to PAR_MMC_TYP After issuing positive online permission, the PLC transfers the HMI parameters to the online interface PAR_MMC_TYP -> MMC1_TYP
DB19 DBB123	MMC1_MSTT_ADR
	refer to PAR_ MSTT_ADR After issuing positive online permission, the PLC transfers the HMI parameters to the online interface PAR_ MSTT_ADR -> MMC1_MSTT_ADR
DB19 DBB124	MMC1_STATUS
	Requests from online HMI to PLC or vice-versa. The meaning of the signal is dependent on MMC1_Z_INFO
OFFL_REQ_PLC (1)	PLC to HMI: PLC wants to displace control unit by offline request.
OFFL_CONF_PLC (2)	HMI to PLC: Acknowledgement of OFFL_REQ_PLC
OFFL_REQ_OP (3)	HMI to PLC: Control unit would like to go offline from this NCU and outputs an offline request
OFFL_CONF_OP (4)	PLC to HMI: Acknowledgement of OFFL_REQ_OP
S_ACT (6)	HMI to PLC: Control unit goes online or changes operating focus
OFFL_REQ_FOC (7)	HMI to PLC: Control unit would like to take operating focus away from this NCU
OFFL_CONF_FOC (8)	PLC to HMI: Acknowledgement of OFFL_REQ_FOC
ONL_REQ_FOC (9)	HMI to PLC: Control unit would like to set operating focus to this NCU
ONL_PERM_FOC (10)	PLC to HMI: Acknowledgement of ONL_REQ_FOC
DB19 DBB125	MMC1_Z_INFO
	Request from online HMI to PLC or vice-versa. The meaning of the signal is dependent on MMC1_STATUS
DISC_FOC (9)	Control unit switches operating focus to another NCU.
OK (10)	Positive acknowledgement
CONNECT (11)	Control unit has gone online on this NCU.
PLC_LOCKED (14)	The control unit switchover disable is set in the HMI-PLC interface. Control unit cannot go offline from this NCU or change operating focus.
PRIO_H (15)	Control units with a higher priority are operating on this NCU. Requesting control unit cannot go online to this NCU

Bit signals

DB19 **DBX 126.0** **Data Block**	**MMC1_SHIFT_LOCK** **Disable/enable control unit switchover**	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Control unit switchover or change in operating focus is disabled. The current control unit-NCU constellation remains unchanged.	
Signal state 0 or edge change 1 → 0	Control unit switchover or change in operating focus is enabled	

DB19 **DBX 126.1** **Data Block**	**MMC1_MSTT_SHIFT_LOCK** **Disable/enable MCP switchover**	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	MCP switchover is disabled. The current MCP-NCU constellation remains unchanged.	
Signal state 0 or edge change 1 → 0	MCP switchover is enabled	

DB19 **DBX 126.2** **Data Block**	**MMC1_ACTIVE_REQ** **Control unit 1 requests active operating mode**	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	HMI to PLC: passive control unit 1 requests active operating mode	
Signal state 0 or edge change 1 → 0	PLC to HMI: Request received	

DB19 DBX 126.3 Data Block	MMC1_ACTIVE_PERM Active/passive operating mode
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	PLC to HMI: passive operator unit 1 can change to active operating mode
Signal state 0 or edge change 1 → 0	PLC to HMI: active operator panel must change to passive operating mode

DB19 DBX 126.4 Data Block	MMC1_ACTIVE_CHANGED Active/passive operating mode of HMI
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	HMI to PLC: Control unit has completed changeover from passive to active mode
Signal state 0 or edge change 1 → 0	HMI to PLC: Control unit has completed changeover from active to passive mode

DB19 DBX126.5 Data Block	MMC1_CHANGE_DENIED Operating mode changeover rejected
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	HMI to PLC or PLC to HMI depending on status of interface: Operating mode cannot be changed due to uninterruptible processes on active control unit.
Signal state 0 or edge change 1 → 0	HMI to PLC or PLC to HMI depending on status of interface: Acknowledgement on MMC1_CHANGE_DENIED (FALSE → TRUE)

2. HMI/PLC online interface

The signals of the 2nd HMI/PLC online interface are analogous in meaning to the signals of the 1st HMI/PLC online interface (MMC2_ ... replaces MMC1_...)..

Sign-of-life monitoring HMI

After a control unit has gone online to an NCU, the HMI sign of life is set in the interface. (E_BTSSReady, E_MMCMPI_Ready, E_MMC2Ready)

The signals are automatically set by the HMI when the control unit goes online and stay set for as long as it remains online.

They are provided separately for each HMI/PLC interface and used by the PLC to monitor the HMI sign of life.

First HMI/PLC online interface

A distinction between an control unit link via the OPI (1.5 Mbaud) or the MPI (187.5 kbaud) is made on this interface.

The signal corresponding to the bus type is set while the control unit is online.

DB10 DBX104.0	MCP1 ready
FALSE	MCP1 is not ready
TRUE	MCP1 is ready
DB10 DBX104.1	MCP2 ready
FALSE	MCP2 is not ready
TRUE	MCP2 is ready
DB10 DBX104.2	HHU ready
FALSE	HHU is not ready
TRUE	HHU is ready
DB10 DBX108.3	E_MMCBTSSReady
FALSE	No control unit online to OPI
TRUE	Control unit online to OPI
DB10 DBX108.2	E_MMCMPIReady
FALSE	No control unit online to MPI
TRUE	Control unit online to MPI

Second HMI/PLC online interface

This interface utilizes a group signal for both bus types. No distinction is made between OPI and MPI.

DB10 DBX108.1	E_MMC2Ready
FALSE	no control unit online to OPI or MPI
TRUE	Control unit online to OPI or MPI

The sign-of-life monitor is switched on by the PLC as soon as a control unit has gone online to its interface and switched off again when it goes offline.

Sign-of-life monitoring will be **enabled:**
as soon as control unit or HMI logs on online to its HMI/PLC interface with S_ACT/CONNECT.

Sign-of-life monitoring will be **disabled:**
as soon as control unit goes offline.

1. HMI wants to switchover and log off from PLC with OFFL_REQ_OP/OK
2. PLC acknowledges to HMI with OFFL_CONF_OP/OK
3. Control unit or HMI will be displaced by PLC with OFFL_REQ_PLC/OK

   HMI acknowledges to PLC with OFFL_CONF_PLC/OK

In both instances the PLC detects that a control unit is going offline and waits for the TRUE-FALSE edge of its HMI sign-of-life signal.

The PLC then ceases to monitor the sign-of-life signal.

18.2.3 Signals from NC (DB10)

DB10 DBX107.6	NCU link active
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	NCU link communication is active.
Signal state 0 or edge change 1 → 0	No NCU link communication is active.
Signal irrelevant for ...	System with an NCU.
References	Device Manual, NCU 7x0.3 PN

18.2.4 Signals from axis/spindle (DB31, ...)

DB31, ... DBX60.1	NCU link axis active	
Edge evaluation:		Signal(s) updated:
Signal state 1 or edge change 0 → 1	Axis is active as NCU link axis.	
Signal state 0 or edge change 1 → 0	Axis is used as a local axis.	
Signal irrelevant for ...	System with an NCU.	
Additional references	NCU 7x0.3 PN Manual	

DB31, ... DBX61.2	Axis ready	
Edge evaluation:		Signal(s) updated:
Meaning	The signal is routed on the NCU in the NCU link group to which the axis is physically connected.	
Signal state 1 or edge change 0 → 1	Axis is ready.	
Signal state 0 or edge change 1 → 0	Axis is not ready. This status will be set when the channel, the operating modes group or the NCK have generated the alarm "not ready".	

DB31, ... DBX62.7	Axis container rotation active	
Edge evaluation:		Signal(s) updated:
Signal state 1 or edge change 0 → 1	An axis container rotation is active for the axis.	
Signal state 0 or edge change 1 → 0	An axis container rotation is not active for the axis.	

18.3 Manual and Handwheel Travel (H1)

18.3.1 Signals from NC (DB10)

<table>
<tr><td>DB10
DBB97, 98, 99</td><td colspan="9">Channel number geometry axis for handwheel 1, 2, 3</td></tr>
<tr><td>Edge evaluation: No</td><td colspan="9">Signal(s) updated: Cyclic</td></tr>
<tr><td rowspan="7">Significance of signal</td><td colspan="9">The operator can assign an axis to the handwheel (1, 2, 3) directly on the operator panel front. If this axis is a geometry axis (IS "Machine axis" = 0), the assigned channel number for the handwheel in question is transferred to the PLC.
In this way, the IS "Activate handwheel" is set for the selected geometry axis in accordance with the state set by the operator (IS "Handwheel selected").
The following codes apply to the channel number:</td></tr>
<tr><td colspan="8">Bit</td><td rowspan="2">Channel number</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td colspan="9">With machine axes (IS "Machine axis" = 1), the IS "Channel number geometry axis for handwheel 1, 2, 3" has no meaning.
For further information, see IS "Axis number for handwheel 1, 2, 3".</td></tr>
<tr><td>Corresponding to ....</td><td colspan="9">DB10 DBB100 ff (axis number for handwheel 1, 2, 3)
DB10 DBX100.6 ff (handwheel selected)
DB10 DBX100.7 ff (machine axis)
DB21, ... DBX12.0 - 12.2 ff (activate handwheel)</td></tr>
<tr><td>Application example(s)</td><td colspan="9">If DB10 DBB97 = 2, then handwheel 1 is assigned to channel 2.</td></tr>
</table>

DB10 DBB100, 101, 102 Bit 0 - 4	**Axis number for handwheel 1, 2 or 3**
Edge evaluation: no	Signal(s) updated: Cyclic
Significance of signal	The operator can assign an axis to every handwheel directly via the operator panel front. To do so, he defines the required axis (e.g. X). The basic PLC program provides the number of the axis plus the information "machine axis or geometry axis" (IS "machine axis") as HMI interface signals. The basic PLC program sets the interface signal "Activate handwheel" for the defined axis. Depending on the setting in the HMI interface signal "machine axis", either the interface for the geometry axis or for the machine axis is used. The following must be noted when assigning the axis designation to the axis number: NST "machine axis" = 1; e.g. machine axis: The assignment is done via the machine data: MD10000 $MN_AXCONF_MACHAX_NAME_TAB[n] (machine axis name). NST "machine axis" = 0; e.g. geometry axis: The assignment is done via the machine data: MD20060 $MC_AXCONF_GEOAX_NAME_TAB[n] (geometry axis name in channel) With the NST "Channel number geometry axis handwheel n" the channel number assigned to the handwheel is defined. For following codes are used for the axis number:
Corresponding to	DB10 DBX97 ff (Channel number geometry axis handwheel n) DB10 DBX100.6 ff (handwheel selected) DB10 DBX100.7 ff (machine axis) DB21, ... DBX12.0 to DBX12.2 ff (activate handwheel) DB31, ... DBX4.0 to DBX4.2 (activate handwheel) MD10000 $MN_AXCONF_MACHAX_NAME_TAB [n] (machine axis name) MD20060 $MC_AXCONF_GEOAX_NAME_TAB [n] (geometry axis in the channel)

Bit					Axis number
4	3	2	1	0	
0	0	0	0	0	–
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
0	0	1	0	1	5
0	0	1	1	0	6
0	0	1	1	1	7
0	1	0	0	0	8

DB10 DBX100.6, 101.6, 102.6	**Handwheel selected (for handwheel 1, 2 or 3)**	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The operator has selected the handwheel for the defined axis via the operator panel front (i.e. activated). This information is made available by the basic PLC program at the HMI interface. The basic PLC program sets the interface signal: DB21, ... DBX12.0-12.2 ff (Activate handwheel) for the defined axis to "1". The associated axis is also displayed at the HMI interface: DB10 DBX100.7 ff (machine axis) and DB10 DBB100 ff (axis number for handwheel 1). As soon as the handwheel is active, the axis can be traversed in JOG mode with the handwheel (DB21, ... DBX40.0-40.2 ff (Handwheel active).	
Signal state 0 or edge change 1 → 0	The operator has disabled the handwheel for the defined axis via the operator panel front. This information is made available by the basic PLC program at the HMI interface. The basic PLC program can set the interface signal: DB21, ... DBX12.0-12.2 ff (Activate handwheel) for the defined axis to "0".	
Corresponding to	DB10 DBB100 ff (axis number) DB10 DBX100.7 ff (machine axis) DB21, ... DBX12.0-12.2 ff (activate handwheel) DB21, ... DBX40.0 - DBX40.2 ff (handwheel active) DB31, ... DBX4.0 - DBX4.2 (activate handwheel) DB10 DBB97 ff (channel number geometry axis for handwheel 1, 2 or 3)	

DB10 DBX100.7, 101.7, 102.7	**Machine axis (for handwheel 1, 2 or 3)**	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The operator has assigned an axis to the handwheel (1, 2, 3) directly on the operator panel front. This axis is a machine axis. For further information see IS "Axis number".	
Signal state 0 or edge change 1 → 0	The operator has assigned an axis to the handwheel (1, 2, 3) directly on the operator panel front. This axis is a geometry axis. For further information see IS "Axis number".	
Corresponding to	DB10 DBB100 ff (axis number) DB10 DBX100.6 ff (handwheel selected) DB10 DBB97 ff (channel number geometry axis for handwheel 1, 2 or 3)	

18.3.2 Signals to channel (DB21, ...)

Overview of signals to channel (to NCK)

DB21, ... DBX0.3	**Activate DRF**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Request to activate the DRF function. With the DRF function, the DRF offset can be changed in the AUTOMATIC and MDI modes using a handwheel.
Signal state 0	No request of the DRF function.
Signal irrelevant for ...	JOG mode
Corresponding to ...	DB21, ... DBX24.3 (DRF selected)

DB21, ... DBX12.0-2, DBX16.0-2, DBX20.0-2	**Handwheel assignment for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state <> 0	Request for activation of the appropriate geometry axis handwheel. The interface can be interpreted either bit or binary-coded. The selection is defined using machine data: MD11324 $MN_HANDWH_VDI_REPRESENTATION
	Bit-coded: Maximum of three handwheels **Note** At any one time, the axis can only be assigned to one handwheel. If several interface signals are set simultaneously, then the following priority applies: "Handwheel 1" before "handwheel 2" before "handwheel 3"

Number of the assigned handwheel	Bit 2	Bit 1	Bit 0
1	0	0	1
2	0	1	0
3	1	0	0

Binary-coded: Maximum of six handwheels

Number of the assigned handwheel	Bit 2	Bit 1	Bit 0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0

Signal state 0	No request for activation of a handwheel.
Corresponding to ...	DB21, ... DBX40.6-7 ff (handwheel active for geometry axis)

DB21, ... DBX12.4, DBX16.4, DBX20.4	**Traversing key disable for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	A traversing request using the "Plus" and "Minus" traversing keys is ignored for the geometry axis. If the traversing key disable is activated while traversing, then traversing is canceled.
Signal state 0	The plus and minus traversing keys are enabled.
Application example(s)	It is thus possible, depending on the operating mode, to disable manual traversing of the geometry axis in JOG mode with the traversing keys from the PLC user program.
Corresponding to ...	DB21, ... DBX12.6-7 ff (traversing key plus or traversing key minus for geometry axis)

DB21, ... DBX12.5, DBX16.5, DBX20.5	**Rapid traverse override for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Moves the axis due to a traversing request: • DB21, ... DBX12.7 == 1 (traversing key plus) • DB21, ... DBX12.6 == 1 (traversing key minus) when the interface signal is set, the geometry axis is moved with rapid traverse. The rapid traverse feedrate is defined in machine data: MD32010 $MA_JOG_VELO_RAPID (conventional rapid traverse) The rapid traverse override is effective in JOG mode for: • Continuous travel • Incremental travel (INC1, INC10, ...) The rapid traverse velocity can be influenced using the rapid traverse override switch.
Signal state 0	The geometry axis traverses with the defined JOG velocity: SD41110 $SN_JOG_SET_VELO (axis velocity with JOG) or MD32020 $MA_JOG_VELO (conventional axis velocity).
Signal irrelevant for ...	• Operating modes AUTOMATIC and MDI • Reference point approach (JOG mode)
Corresponding to ...	DB21, ... DBX12.6-7 ff (traversing key plus and traversing key minus for geometry axis)
Further references	Function Manual, Basic Function; Feedrates (V1)

DB21, ... DBX12.6-7, DBX16.6-7, DBX20.6-7	**Plus and minus traversing keys for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The selected geometry axis can be traversed in both directions in JOG mode with the traversing keys plus and minus. Depending on the active machine function as well as the setting "JOG or Continuous mode": SD41050 $SN_JOG_CONT_MODE_LEVELTRIGGRD (JOG/Cont. mode continuous with JOG) and MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD (INC and REF in JOG mode) different reactions are triggered on a signal change. 1. Continuous traversing with JOG mode The geometry axis traverses in the direction concerned as long as the interface signal is set to 1 (and as long as the axis position has not reached an activated limitation). 2. Continuous traversing with Continuous mode On the first edge change 0 →1, the geometry axis starts to traverse in the relevant direction. This traversing movement still continues when the edge changes from 1 → 0. Any new edge change 0 →1 (same traversing direction!) stops the traversing movement. 3. Incremental traversing with JOG mode With signal 1 the geometry axis starts to traverse at the set increment. If the signal changes to the 0 state before the increment is traversed, the traversing movement is interrupted. When the signal state changes to 1 again, the movement is continued. The geometry axis can be stopped and started several times as described above until it has traversed the complete increment. 4. Incremental traversing with Continuous mode On the first edge change 0 → 1 the geometry axis starts to traverse at the set increment. If another edge change 0 → 1 is performed with the same traverse signal before the geometry axis has traversed the increment, the traversing movement will be cancelled. The increment traversing will then not be completed. **Note** • If both traversing signals (plus and minus) are set at the same time there is no traversing or the current traversing is aborted. • In contrast to machine axes, for geometry axes, only one geometry axis can be traversed at any one time using the traversing keys. • Traversing by means of the traversing keys can be locked via DBX12.4 = 1 ff. (traversing key disable).
Signal state 0	See cases 1 to 4 above.
Signal irrelevant for ...	Operating modes AUTOMATIC and MDI
Special cases, errors, ...	The geometry axis cannot be traversed in JOG mode: • If it is already being traversed via the axial PLC interface (as a machine axis). • If another geometry axis is already being traversed with the traversing keys. Alarm 20062 "Axis already active" is output.
Corresponding to ...	DB31, ... DBX8.7 or DBX8.6 (traversing keys plus and minus for machine axes) DB21, ... DBX12.4 ff (traversing key disable for geometry axes)

DB21, ... DBX13.0-5, DBX17.0-5, DBX21.0-5	**Request for incremental machine function for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Request for a machine function for incremental traversing of the geometry axis in JOG mode: • Bit 0 = INC1 • Bit 1 = INC10 • Bit 2 = INC100 • Bit 3 = INC1000 • Bit 4 = INC10000 • Bit 5 = INCvar An increment corresponds to an actuation of one of the "plus" and "minus" traversing keys, or a detent position of the active handwheel. Specification of the increment sizes via: • INC1 to INC10000: MD11330 $MN_JOG_INCR_SIZE_TAB (increment size for INC/handwheel) • INCvar: SD41010 $SN_JOG_VAR_INCR_SIZE (size of the variable increment for JOG) **Note** If several requests are set simultaneously, no machine function becomes active.
Signal state 0	No machine function is requested. If a geometry axis is currently being traversed via a machine function, the movement is aborted through deselection or change of the machine function.
Corresponding to ...	DB21, ... DBB41 ff (active machine function INC1,...) for geometry axes DB21, ... DBX13 ff (machine function continuous) for geometry axes

DB21, ... DBX13.6, DBX17.6, DBX21.6	**Request for continuous machine function for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The machine function for continuous traversing of the geometry axis in JOG mode with the traversing keys "Plus" and "Minus" is requested.
Signal state 0	Machine function "Continuous traversing" is not requested.
Corresponding to ...	DB21, ... DBB41.6 ff (active continuous machine function)

DB21, ... DBX15.0, DBX 19.0, DBX 23.0	**Handwheel direction of rotation inversion for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Request to invert the handwheel direction of rotation. It is only permissible to change the interface signal when the geometry axis is at a standstill.
Signal state 0	The handwheel direction of rotation to which geometry axis 1, 2 or 3 is assigned, is not inverted.
Application example(s)	• The handwheel direction of movement does not match the expected direction of the axis. • A handwheel (HT2, HT8) has been assigned to various axes.
Corresponding to ...	DB21, ... DBX43.0, 49.0, 55.0 (handwheel direction of rotation inversion active for geometry axis 1, 2, 3)

DB21, ... DBX30.0-2	**Activate contour handwheel**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Request for activation of the corresponding handwheel for the "Contour handwheel / path specification with handwheel" function. The interface can be interpreted either bit or binary-coded. The specification is performed via: MD11324 $MN_HANDWH_VDI_REPRESENTATION
	Bit-coded: Maximum of three handwheels **Note** At any one time, the axis can only be assigned to one handwheel. If several interface signals are set simultaneously, then the following priority applies: "Handwheel 1" before "handwheel 2" before "handwheel 3"

Number of the assigned handwheel	Bit 2	Bit 1	Bit 0
1	0	0	1
2	0	1	0
3	1	0	0

Binary-coded: Maximum of six handwheels

Number of the assigned handwheel	Bit 2	Bit 1	Bit 0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0

Signal state 0	No request for activation of a handwheel.
Corresponding to ...	DB21, ... DBX37.0-2 (contour handwheel active)

DB21, ... DBX320.0-2, DBX324.0-2, DBX328.0-2	**Activate handwheel for orientation axis (1, 2, 3)**			
Edge evaluation: No	Signal(s) updated: Cyclically			
Signal state 1	Request for activation of the corresponding handwheel for the orientation axis. The interface can be interpreted either bit or binary-coded. The specification is performed via: MD11324 $MN_HANDWH_VDI_REPRESENTATION			
	Bit-coded: Maximum of three handwheels **Note** At any one time, the axis can only be assigned to one handwheel. If several interface signals are set simultaneously, then the following priority applies: "Handwheel 1" before "handwheel 2" before "handwheel 3"			
	Number of the assigned handwheel	**Bit 2**	**Bit 1**	**Bit 0**
	1	0	0	1
	2	0	1	0
	3	1	0	0
	Binary-coded: Maximum of six handwheels			
	Number of the assigned handwheel	**Bit 2**	**Bit 1**	**Bit 0**
	1	0	0	1
	2	0	1	0
	3	0	1	1
	4	1	0	0
	5	1	0	1
	6	1	1	0
Signal state 0	No request for activation of a handwheel.			
Corresponding to ...	DB21, ... DBX332.Bit 0-2 ff. (handwheel active for orientation axis (1, 2, 3))			

DB21, ... DBX323.0, DBX327.0, DBX331.0	**Handwheel direction of rotation inversion for orientation axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Request for inversion of the handwheel direction of rotation assigned to orientation axis 1, 2 or 3. **Note** It is only permissible to change the inversion signal at standstill.
Signal state 0	Inversion has not been requested.
Application example(s)	• The handwheel direction of rotation should match the axis direction of motion. • A handwheel is assigned to several axes with different orientations.
Corresponding to ...	DB21, ... DBX335.0 ff. (handwheel direction of rotation inversion active for orientation axis 1, 2, 3)

18.3.3 Signals from channel (DB21, ...)

Description of signals from channel to PLC

DB21, ... DBX24.3	**DRF selected**	
Edge evaluation: No		Signal(s) updated: Cyclically
Signal state 1	The DRF function is active.	
Signal state 0	The DRF function is not active.	
Signal irrelevant for ...	JOG mode	
Corresponding to ...	DB21, ... DBX0.3 (activate DRF)	

DB21, ... DBX33.3	**Handwheel override active**	
Edge evaluation: No		Signal(s) updated: Cyclically
Signal state 1	The function "Handwheel override in Automatic mode" is active for the programmed path axes. The handwheel pulses of the 1st geometry axis function as a velocity override on the programmed path feedrate. In the following cases, the override is inactive: • The path axes have reached the programmed target position • The distance-to-go has been deleted: DB21, ... DBX6.2 == 1 (delete distance-to-go) • RESET was initiated	
Signal state 0	The "Handwheel override in Automatic mode" function is not active.	

DB21, ... DBX37.0-2	**Contour handwheel active**	
Edge evaluation: No		Signal(s) updated: Cyclically
Signal state 1	Feedback signal indicating which handwheel is active for the "Contour handwheel/path input using handwheel". The interface can be interpreted either bit or binary-coded. The specification is performed via: MD11324 $MN_HANDWH_VDI_REPRESENTATION	
	Bit-coded: Maximum of three handwheels **Note** At any one time, the axis can only be assigned to one handwheel. If several interface signals are set simultaneously, then the following priority applies: "Handwheel 1" before "handwheel 2" before "handwheel 3"	

Number of the assigned handwheel	Bit 2	Bit 1	Bit 0
1	0	0	1
2	0	1	0
3	1	0	0

<table>
<tr><th>DB21, ...
DBX37.0-2</th><th colspan="5">Contour handwheel active</th></tr>
<tr><td rowspan="8"></td><td colspan="5">Binary-coded: Maximum of six handwheels</td></tr>
<tr><td rowspan="7"></td><td>Number of the assigned handwheel</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>2</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>3</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>4</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>5</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>6</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>Signal state 0</td><td colspan="5">The "Contour handwheel/path input using handwheel" is not assigned to a handwheel.</td></tr>
<tr><td>Corresponding to ...</td><td colspan="5">DB21, ... DBX30.0-2 (handwheel assignment for contour handwheel)</td></tr>
</table>

<table>
<tr><th>DB21, ...
DBX40.0-2,
DBX46.0-2,
DBX52.0-2</th><th colspan="5">Handwheel active for geometry axis (1, 2, 3)</th></tr>
<tr><td colspan="2">Edge evaluation: No</td><td colspan="4">Signal(s) updated: Cyclically</td></tr>
<tr><td rowspan="16">Signal state 1</td><td colspan="5">Feedback signal indicating which handwheel is active for the geometry axis.
The interface can be interpreted either bit or binary-coded. The specification is performed via:
MD11324 $MN_HANDWH_VDI_REPRESENTATION</td></tr>
<tr><td colspan="5">Bit-coded: Maximum of three handwheels
Note
At any one time, the axis can only be assigned to one handwheel. If several interface signals are set simultaneously, then the following priority applies: "Handwheel 1" before "handwheel 2" before "handwheel 3"</td></tr>
<tr><td rowspan="4"></td><td>Number of the assigned handwheel</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>2</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>3</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td colspan="5">Binary-coded: Maximum of six handwheels</td></tr>
<tr><td rowspan="7"></td><td>Number of the assigned handwheel</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>2</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>3</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>4</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>5</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>6</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>Signal state 0</td><td colspan="5">None is active for the geometry axis.</td></tr>
<tr><td>Corresponding to ...</td><td colspan="5">DB21, ... DBX12.0-2 ff (activate handwheel)</td></tr>
</table>

DB21, ... DBX40.4-5, DBX46.4-5, DBX52.4-5	**Plus or minus traversing request for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	A traversing request is available for the geometry axis for the corresponding traversing direction. • Bit 4 = minus traversing request • Bit 5 = plus traversing request Operating modes: • JOG mode: Plus or minus traversing key • REF mode: Traversing key that initiates a traversing movement in the direction of the reference point. • AUTOMATIC/MDI modes: A program block with a traversing operation is executed for a geometry axis.
Signal state 0	There is no traversing request available for the geometry axis.
Corresponding to ...	DB21, ... DBX40.7 or DBX40.6 DB21, ... DBX46.7 or DBX46.6 DB21, ... DBX52.7 and/or DBX52.6 (traversing command plus and minus)

DB21, ... DBX40.7-6, DBX46.7-6, DBX52.7-6	**Traversing command plus and minus for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Description	The output of the drive commands depends on: MD17900 $MN_VDI_FUNCTION_MASK, bit 0 **Bit 0** – **Meaning** 0 – The drive commands are already output when a traversing request is active. 1 – The drive commands are only output when the axis traverses.
Signal state 1	Request to traverse the geometry axis in the corresponding direction. • Bit 6 = minus drive command • Bit 7 = plus drive command The response to a drive command depends on the current operating mode: • JOG mode: Traverse the axis in the traversing direction plus or minus. • REF mode: Traverse the axis only in the direction of the reference point. • AUTOMATIC/MDI mode: A block containing a position for the axis is executed.
Signal state 0	There is no traversing request available for the geometry axis.

DB21, ... DBX40.7-6, DBX46.7-6, DBX52.7-6	**Traversing command plus and minus for geometry axis (1, 2, 3)**
Application example(s)	Releasing the axis clamp when the traversing command is identified. **Note** For axes on which the clamping is not released until a drive command is detected, continuous-path mode (`G64`) is not possible.
Corresponding to ...	DB21, ... DBX12.7 or DBX12.6 ff (traversing key plus and minus for geometry axis) DB21, ... DBX 40, 46, 52 Bit 5 (traversing request plus/minus)

DB21, ... DBX41.0-6, DBX47.0-6, DBX53.0-6	**Active machine functions for geometry axis (1, 2, 3) INC1, INC10, INC100, INC1000, INC10000, INCvar, continuous**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The corresponding machine function is active. • Bit 0 = 1 INC • Bit 1 = 10 INC • Bit 2 = 100 INC • Bit 3 = 1000 INC • Bit 4 = 10000 INC • Bit 5 = Var. INC • Bit 6 = continuous The reaction to actuation of the traversing key or rotation of the handwheel varies, depending on which machine function is active.
Signal state 0	The machine function in question is not active.
Corresponding to ...	DB21, ... DBB13 bit 0-5 ff (machine function INC1, ... for geometry axis) DB21, ... DBB13 bit 6 ff (machine function continuous for geometry axis)

DB21, ... DBX43.0, 49.0, 55.0	**Handwheel direction of rotation inversion active for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The inversion of the handwheel direction of rotation is active for the geometry axis.
Signal state 0	The inversion of the handwheel direction of rotation is not active for the geometry axis.
Corresponding to ...	DB31, ... DBX15.0, DBX19.0, DBX23.0.(invert handwheel direction of rotation for geometry axis)

DB21, ... DBX332.0-2, DBX336.0-2, DBX340.0-2	**Handwheel active for orientation axis (1, 2, 3)**			
Edge evaluation: No	Signal(s) updated: Cyclically			
Signal state 1	Feedback signal indicating which handwheel is active for the orientation axis. The interface can be interpreted either bit or binary-coded. The specification is performed via: MD11324 $MN_HANDWH_VDI_REPRESENTATION			
	Bit-coded: Maximum of three handwheels **Note** At any one time, the axis can only be assigned to one handwheel. If several interface signals are set simultaneously, then the following priority applies: "Handwheel 1" before "handwheel 2" before "handwheel 3"			
	Number of the assigned handwheel	**Bit 2**	**Bit 1**	**Bit 0**
	1	0	0	1
	2	0	1	0
	3	1	0	0
	Binary-coded: Maximum of six handwheels			
	Number of the assigned handwheel	**Bit 2**	**Bit 1**	**Bit 0**
	1	0	0	1
	2	0	1	0
	3	0	1	1
	4	1	0	0
	5	1	0	1
	6	1	1	0
Signal state 0	A handwheel is not active for the orientation axis.			
Corresponding to ...	DB21, ... DBX332.0-2 ff (activate handwheel)			

DB21, ... DBX332.4-5, DBX336.4-5, DBX340.4-5	**Plus and minus traversing request for orientation axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
The signal is the same as the previous traversing command signal.	
Signal state 1	• JOG mode: With the plus or minus traversing key. • REF mode: With the traversing key that takes the axis to the reference point. • AUTOMATIC/MDI mode: A program block containing a coordinate value for the axis in question is executed.

DB21, ... DBX332.4-5, DBX336.4-5, DBX340.4-5	**Plus and minus traversing request for orientation axis (1, 2, 3)**
Signal state 0	A traversing command in the relevant axis direction has not been given or a traversing movement has been completed. • JOG mode: The traversing command is reset depending on the current setting "JOG or Continuous mode" (see DBX12.7 or DBX12.6 ff. While traversing with the handwheel. • REF mode: When the reference point is reached. • AUTOMATIC/MDI mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. interface signal DB21, … DBX25.7 (axis disabled) is active.
Corresponding to ...	DB31, ... DBX332.7 or DBX332.6 DB31, ... DBX336.7 or DBX336.6 DB31, ... DBX340.7 and/or DBX340.6 (traversing command plus and minus)

<table>
<tr><th>DB21, ...
DBX332.6-7,
DBX336.6-7,
DBX340.6-7</th><th colspan="2">Traversing command plus and minus for orientation axis (1, 2, 3)</th></tr>
<tr><td colspan="2">Edge evaluation: No</td><td>Signal(s) updated: Cyclically</td></tr>
<tr><td colspan="3">The interface signal behaves differently depending on following machine data setting:
• MD17900 $MN_VDI_FUNCTION_MASK, bit 0 == 0
Behavior corresponding to the following description:
• MD17900 $MN_VDI_FUNCTION_MASK, bit 0 == 1
Signal state 1: Only if the geometry axis actually traverses.
The interface signal
DB21, ... DBX 332, 336, 340 Bit 5, 4 (traversing request plus/minus)
which is always output, has the same effect as signal traversing command plus/minus with MD17900, bit 0 = 0.</td></tr>
<tr><td>Signal state 1</td><td colspan="2">A traverse movement of the axis is to be executed in one or the other direction. Depending on the mode selected, the command is triggered in different ways:
• JOG mode:
With the plus or minus traversing key.
• REF mode:
With the traversing key that takes the axis to the reference point.
• AUTOMATIC/MDI mode:
A program block containing a coordinate value for the axis in question is executed.</td></tr>
</table>

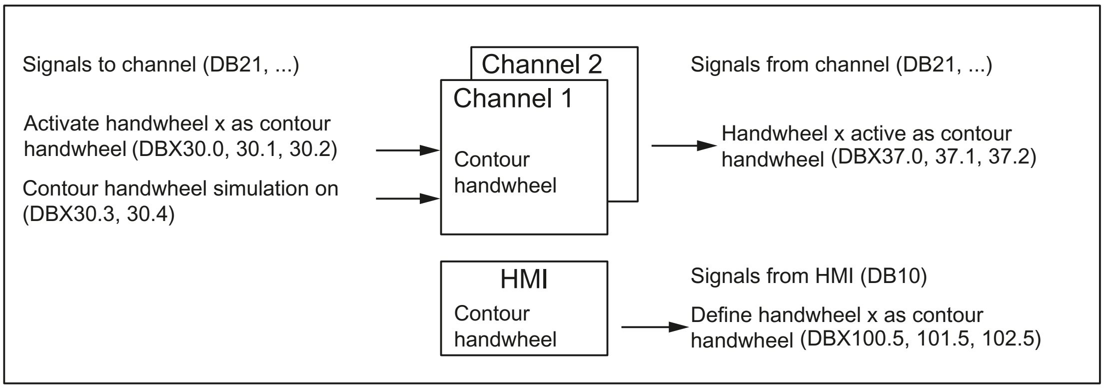
DB21, ... DBX332.6-7, DBX336.6-7, DBX340.6-7	Traversing command plus and minus for orientation axis (1, 2, 3)
Signal state 0	A traversing command in the relevant axis direction has not been given or a traversing movement has been completed. • JOG mode: The traversing command is reset depending on the current setting "JOG or Continuous mode" (see DB21, ... DBX12.7 or DBX12.6 ff). While traversing with the handwheel. • REF mode: When the reference point is reached. • AUTOMATIC/MDI mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. interface signal DB21, … DBX25.7 (axes disable) is active.
Application example(s)	To release clamping of axes with clamping (e.g. on a rotary table). **Note:** If the clamping is not released until the traversing command is given, these axes cannot be operated under continuous path control!
Corresponding to ...	DB21, ... DBX12.7 and/or DBX12.6 ff (traversing key plus and minus for geometry axis) DB21, ... DBX 332, 336, 340 Bit 5, 4 (traversing request plus/minus)

DB21, ... DBX377.4	JOG retract active	
Edge evaluation: No		Signal(s) updated: Cyclically
Signal state 1	JOG retract has been selected and is active.	
Signal state 0	JOG retract has **not** been selected.	
Corresponding to ...	DB21, ... DBX377.5 (JOG retract: Retraction data available)	

DB21, ... DBX377.5	Retraction data available	
Edge evaluation: No		Signal(s) updated: Cyclically
Signal state 1	JOG retract: Retraction data is available for traversing in the tool direction. JOG retract can be selected (user interface or PI service "RETRAC").	
Signal state 0	JOG retract: Retraction data is **not** available for traversing in the tool direction. JOG retract **cannot** be selected	
Corresponding to ...	DB21, ... DBX377.4 (JOG retract active)	

18.3.4 Signals with contour handwheel

Overview of interface signals for contour handwheel



Description of interface signals for contour handwheel

DB10 **DBX100.5** **DBX101.5** **DBX102.5**	**Define handwheel 1 as contour handwheel** **Define handwheel 2 as contour handwheel** **Define handwheel 3 as contour handwheel**	
Edge evaluation: No	Signal(s) updated: Cyclic	
Description	These signals show which handwheel is defined as contour handwheel via the HMI:	
	Signal = 1	Handwheel x is defined as contour handwheel via the HMI.
	Signal = 0	Handwheel x is not defined as contour handwheel.
	In order for the handwheel defined via HMI to become effective as contour handwheel, the corresponding signal has to be combined on interface signal: DB21, ... DBX30.0, 30.1, 30.2 (activate handwheel x as contour handwheel) .	
Special cases, errors, ...	Depending on the settings of parameter HWheelMMC in FB1 of the basic PLC program, these signals are either supplied by the basic program or must be supplied by the PLC user program.	
Corresponding to ...	DB21 ... DBX30.0, 30.1, 30.2 (activate handwheel x as contour handwheel) FB1 parameters HWheelMMC	

DB21, ... **DBX30.0** **DBX30.1** **DBX30.2**	**Activate handwheel 1 as contour handwheel;** **Activate handwheel 2 as contour handwheel;** **Activate handwheel 3 as contour handwheel**	
Edge evaluation: No	Signal(s) updated: Cyclic	
Description	One of the three handwheels can be selected/deselected as contour handwheel via these signals:	
	Signal = 1	Handwheel x is selected as contour handwheel
	Signal = 0	Handwheel x is deselected as contour handwheel

DB21, ... DBX30.0 DBX30.1 DBX30.2	**Activate handwheel 1 as contour handwheel;** **Activate handwheel 2 as contour handwheel;** **Activate handwheel 3 as contour handwheel**
	Enabling/disabling of the contour handwheel can be performed in the middle of a block. Upon enabling, the movement is first decelerated and then traversed according to the contour handwheel. Upon disabling, the movement is decelerated and the NC program is continued immediately. If the NC program is to be continued only after a new NC Start, then deactivation of the contour handwheel in the PLC user program must be combined with an NC Stop.
Special cases, errors, ...	The signal is kept beyond an NC Reset.
Corresponding to ...	DB21, ... DBX37.0, 37.1, 37.2 (handwheel x active as contour handwheel)

<table>
<tr><th>DB21, ...
DBX30.3
DBX30.4</th><th colspan="3">Simulation contour handwheel on
Negative direction simulation contour handwheel</th></tr>
<tr><td colspan="2">Edge evaluation: No</td><td colspan="2">Signal(s) updated: Cyclic</td></tr>
<tr><td rowspan="7">Description</td><td colspan="3">For enabling/disabling simulation of the contour handwheel, and for definition of the traversing direction, these signals have to be set as follows:</td></tr>
<tr><td>Bit 3</td><td>Bit 4</td><td>Meaning</td></tr>
<tr><td>0</td><td>0</td><td>Simulation off</td></tr>
<tr><td>0</td><td>1</td><td>Simulation off</td></tr>
<tr><td>1</td><td>0</td><td>Simulation On, direction as programmed</td></tr>
<tr><td>1</td><td>1</td><td>Simulation On, direction opposite programmed direction</td></tr>
<tr><td colspan="3">During simulation, the feedrate is no longer defined by the contour handwheel, but traversing occurs with the programmed feedrate on the contour.
When the function is deselected, the current movement is decelerated by the braking ramp. When the traversing direction is switched, the current movement is decelerated by the braking ramp, and traversing occurs in the opposite direction.</td></tr>
<tr><td>Special cases, errors, ...</td><td colspan="3">Simulation is only effective in AUTOMATIC mode and can only be activated when the contour handwheel is activated.</td></tr>
</table>

<table>
<tr><th>DB21, ...
DBX31.5</th><th colspan="2">Invert handwheel direction of rotation for contour handwheel</th></tr>
<tr><td colspan="2">Edge evaluation: No</td><td>Signal(s) updated: Cyclic</td></tr>
<tr><td>Description</td><td colspan="2">You can invert the direction of rotation of a contour handwheel by setting this PLC interface signal.</td></tr>
<tr><td>Application example(s)</td><td colspan="2">• The direction of movement of the handwheel does not match the expected direction of the axis.
• A handwheel (HT2, HT8) has been assigned to various axes.</td></tr>
<tr><td>Special cases, errors, ...</td><td colspan="2">It is only permissible to change the inversion signal at standstill.</td></tr>
<tr><td>Corresponding to ...</td><td colspan="2">DB31, ... DBX39.5 (handwheel direction of rotation inversion active for contour handwheel)</td></tr>
</table>

DB21, ... DBX37.0 DBX37.1 DBX37.2	 Handwheel 1 active as contour handwheel Handwheel 2 active as contour handwheel Handwheel 3 active as contour handwheel	
Edge evaluation: No		Signal(s) updated: Cyclic
Description	These signals show which handwheel is selected as contour handwheel:	
	Signal = 1	Handwheel x is selected as contour handwheel.
	Signal = 0	Handwheel x is deselected as contour handwheel.
Special cases, errors, ...	The signal is kept beyond an NC Reset.	
Corresponding to ...	DB21, ... DBX30.0, 30.1, 30.2 (handwheel x active as contour handwheel)	

DB21, ... DBX39.5	 Handwheel direction of rotation inversion active for contour handwheel	
Edge evaluation: No		Signal(s) updated: Cyclic
Description	This signal indicates whether the direction of rotation was inverted for the contour handwheel:	
	Signal = 1	The direction of rotation of the contour handwheel is inverted.
	Signal = 0	The direction of rotation of the contour handwheel is not inverted.
Corresponding to ...	DB31, ... DBX31.5 (invert handwheel direction of rotation for the contour handwheel)	

18.3.5 Signals to axis/spindle (DB31, ...)

Description of signals to axis/spindle

DB31, ... DBB4 Bit 0-2	 Activate handwheel (1 to 3)	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	This PLC interface signal defines whether this machine axis is assigned to handwheel 1, 2, 3 or no handwheel. Only one handwheel can be assigned to an axis at any one time. If several interface signals: DB31, ... DBX4.0, 4.1, 4.2 (Activate handwheel) are set, priority "Handwheel 1" before "Handwheel 2" before "Handwheel 3" applies. If the assignment is active, the machine axis can be traversed with the handwheel in JOG mode or a DRF offset can be generated in AUTOMATIC or MDA mode.	
Signal state 0 or edge change 1 → 0	Neither handwheel 1, 2 nor 3 is assigned to this geometry axis.	
Application example(s)	The PLC user program can use this interface signal to disable the influence of turning the handwheel on the axis.	
Corresponding to ...	DB31, ... DBX64.0 to DBX64.2 (Handwheel active)	

DB31, ... DBX4.4	Traversing key lock	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The traverse keys plus and minus have no effect on the machine axes in question. It is thus not possible to traverse the machine axis in JOG with the traverse keys on the machine control panel. If the traverse key disable is activated during a traverse movement, the machine axis is stopped.	
Signal state 0 or edge change 1 → 0	The plus and minus traverse keys are enabled.	
Application example(s)	It is thus possible, depending on the operating mode, to disable manual traverse of the machine axis in JOG mode with the traverse keys from the PLC user program.	
Corresponding to ...	DB31, ... DBX4.7 and/or DBX4.6 (traversing key plus and traversing key minus)	

DB31, ... DBX4.5	Rapid traverse override	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	If together with interface signal: DB31, ... DBX4.0, 4.1, 4.2 (traverse key plus or traverse key minus) the PLC interface signal: DB31, ... DBX4.5 (Rapid traverse override) is sent, then the corresponding machine axis is operating with rapid traverse. The rapid traverse feedrate is defined in machine data: MD32010 $MA_JOG_VELO_RAPID (Conventional rapid traverse) . The rapid traverse override is effective in the JOG mode for the following versions: • Continuous jogging • Incremental jogging If rapid traverse override is active, the velocity can be modified with the rapid traverse override switch.	
Signal state 0 or edge change 1 → 0	The machine axis traverses with the defined JOG velocity: SD41110 $SN_JOG_SET_VELO (Axis velocity with JOG) or MD32020 $MA_JOG_VELO (Conventional axis velocity).	
Signal irrelevant for ...	Operating modes AUTOMATIC and MDA Reference point approach (JOG mode)	
Corresponding to ...	DB31, ... DBX4.7 and/or DBX4.6 (traversing key plus and traversing key minus) DB31, ... DBB0 (axial feed/spindle override)	

DB31, ... DBB4 Bit 7, 6	**Plus and minus traverse keys**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1	The selected machine axis can be traversed in both directions in JOG mode using the traversing keys "plus" and "minus". Depending on the active machine function, as well as the settings: • JOG (continuous) SD41050 $SN_JOG_CONT_MODE_LEVELTRIGGRD (jog/ continuous operation for JOG continuous) • JOG-INC (INC and REF in the jog mode) MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD for a signal change, different responses are initiated. **Case 1: Continuous traversal in jog mode** The machine axis traverses in the direction concerned as long as the interface signal is set to 1 (and as long as the axis position has not reached an activated limitation). **Case 2:Continuous traversal in continuous mode** On the first edge change 0 →1 the machine axis starts to traverse in the relevant direction. This traversing movement still continues when the edge changes from 1 → 0. Any new edge change 0 →1 (same traversing direction!) stops the traversing movement. **Case 3:Incremental traversal in jog mode** With signal 1 the machine axis starts to traverse at the set increment. If the signal changes to the 0 state before the increment is traversed, the traversing movement is interrupted. When the signal state changes to 1 again, the movement is continued. The axis can be stopped and started several times as described above until it has traversed the complete increment. **Case 4: Incremental traversal in continuous mode** On the first edge change 0 → 1 the machine axis starts to traverse at the set increment. If another edge change 0 → 1 is performed with the same traverse signal before the axis has traversed the increment, the traversing movement will be cancelled. The increment traversing will then not be completed. If both traverse signals (plus and minus) are set at the same time there is no movement or a current movement is aborted. The effect of the traverse keys can be disabled for every machine axis individually with the PLC interface signal: DB31, ... DBX4.4 (Traverse key disable) .
Signal state 0	See cases 1 to 4 above.
Signal irrelevant for ...	Operating modes AUTOMATIC and MDA
Application example(s)	The machine axis cannot be traversed in JOG mode if it is already being traversed via the channel-specific PLC interface (as a geometry axis). Alarm 20062 is signaled.
Special cases, ...	Indexing axes
Corresponding to ...	DB21, ... DBX12.7, DBX12.6 ff (Traverse keys plus and minus for geometry axes) DB31, ... DBX4.4 (traversing key disable)

DB31, ... DBB5 Bit 0-5	Machine function INC1, INC10, INC100, INC1000, INC10000, INCvar
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1	Request to activate a machine function for incremental traversing of the axis: • Bit 0: INC1 • Bit 1: INC10 • Bit 2: INC100 • Bit 3: INC1000 • Bit 4: INC10000 • Bit 5: INCvar One increment corresponds to actuating the traversing key or a detent position of the handwheel. The size of an increment is defined in the following system data: • INC1 to INC10000: MD11330 $MN_JOG_INCR_SIZE_TAB (increment size for INC/handwheel) • INCvar: SD41010 $SN_JOG_VAR_INCR_SIZE (size of the variable increment for JOG) The feedback signal indicating that the machine function has been activated is realized via: DB31, ... DBB65 (machine function INC1, ...) **Notice** If several bits are simultaneously set, then no machine function is active in the control.
Signal state 0	The corresponding machine function is not requested. If the axis is presently traversing an increment, motion is canceled when the machine function is either selected or changed over.
Corresponding to ...	DB31, ... DBB65 (machine function INC1, ...) DB31, ... DBX5.6 (Machine function continuous)

DB31, ... DBX5.6	Continuous machine function
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1	The machine axis can be continuously traversed with the traversing keys "plus" and "minus" in the JOG mode.
Signal state 0	The machine function "Continuous jogging" is not selected.
Corresponding to ...	DB31, ... DBB65 (active machine function INC1, ..., continuous) DB31, ... DBB5 (machine function INC1, ..., INC10000)

DB31, ... DBX7.0	Invert handwheel direction of rotation (machine axes)
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1	The direction of rotation of the handwheel, which is assigned to the machine axes, is inverted. It is only permissible to change the inversion signal at standstill.

DB31, ... DBX7.0	Invert handwheel direction of rotation (machine axes)
Signal state 0	The handwheel direction of rotation is not inverted.
Application example(s)	• The direction of movement of the handwheel does not match the expected direction of the axis. • The handwheel is assigned to different axes with different orientations.
Corresponding to ...	DB31, ... DBX67.0 (handwheel direction of rotation inversion active for machine axes)

DB31, ... DBB13 Bit 0-2	JOG - approach fixed point
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1	Activates the "approaching fixed point JOG" function. The number of the fixed point to be approached is specified in bits 0-2 in binary code. The selected machine axis can be traversed to the corresponding fixed point with the traverse keys or the handwheel as soon as the function is active (see DB31, ... DBX75.0-2). The fixed points are defined using the following machine data: MD30600 $MA_FIX_POINT_POS[n]
Signal state 0	Deactivates the "approaching fixed point JOG" function.
Corresponding to ...	DB31, ... DBX75.0-2 (JOG - Approach fixed point) DB31, ... DBX75.3-5 (JOG - Approach fixed point) MD30600 $MA_FIX_POINT_POS[n] (fixed value positions of the axis)

18.3.6 Signals from axis/spindle (DB31, ...)

Description of signals from axis/spindle

DB31, ... DBX62.1	**Handwheel override active**	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The function "Handwheel override in AUTOMATIC mode" is active for the programmed positioning axis (FDA[AXi]). Handwheel pulses for this axis affect the programmed axis feedrate either as path definition (FDA=0) or as velocity override (FDA > 0). The interface signal will also be set if "Handwheel override in automatic mode" is active with a concurrent positioning axis with FC18 (for 840D sl).	
Signal state 0 or edge change 1 → 0	The function "Handwheel override in AUTOMATIC mode" is not active for the programmed positioning axis (or concurrent positioning axis). An active handwheel override is not active if: • the positioning axis has reached the target position • the distance-to-go is deleted by axis-specific interface signal DB31, ... DBX2.2 (delete distance-to-go). • a RESET is performed.	

DB31, ... DBB64 Bit 0-2	**Handwheel active (1 to 3)**	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	These PLC interface signals provide feedback whether the machine axis is assigned to handwheel 1, 2, 3 or no handwheel. Only one handwheel can be assigned to an axis at any one time. If several interface signals: DB31, ... DBX4.0 to DBX4.2 (Activate handwheel) are set, priority "Handwheel 1" before "Handwheel 2" before "Handwheel 3" applies. If the assignment is active, the machine axis can be traversed with the handwheel in JOG mode or a DRF offset can be generated in AUTOMATIC or MDA mode.	
Signal state 0 or edge change 1 → 0	Neither handwheel 1, 2 nor 3 is assigned to this geometry axis.	
Corresponding to ...	DB31, ... DBX4.0 to DBX4.2 (activate handwheel) DB10 DBB100.6 ff (handwheel selected)	

DB31, ... DBB64 Bit 5, 4	Plus and minus traversing request	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	A traverse movement of the axis is to be executed in one or the other direction. Depending on the mode selected, the command is triggered in different ways: • JOG mode: With the plus or minus traverse key. • REF mode: With the traverse key that takes the axis to the reference point. • AUT/MDA mode: A program block containing a coordinate value for the axis in question is executed.	
Signal state 0 or edge change 1 → 0	A traversing command in the relevant axis direction has not been given or a traverse movement has been completed. • JOG mode: The traversing command is reset depending on the current setting "jog or continuous mode" (see interface signal DB31, ... DBX4.7 or DBX4.6). While traversing with the handwheel. • REF mode: When the reference point is reached. • AUT/MDA mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. DB21, … DBX25.7 (axes disable) is active.	
Application example(s)	To release clamping of axes with clamping (e.g. on a rotary table). **Note:** If the clamping is not released until the traversing command is given, these axes cannot be operated under continuous path control!	
Corresponding to ...	DB31, ... DBX4.7 and/or DBX4.6 (traversing key plus and traversing key minus) DB31, ... DBX64.7 and/or DBX64.6 (Traversing command plus and minus)	

DB31, ... DBB64 Bit 7, 6	Plus and minus traversing command	
Edge evaluation: no		Signal(s) updated: Cyclic
The signal has the effect as described, if Bit 0 in the machine data: MD17900 $MN_VDI_FUNCTION_MASK (setting for VDI signals) is set to 0. If bit 0 in the MD is set to 1, then the signal changes to 1 only if the axis is actually moving. The interface signal DB31, ... DBX64 Bit 5, 4 (traversing request plus/minus) , which is always output, has the same effect as signal traversing command plus/minus when MD17900 bit 0 = 0.		

DB31, ... DBB64 Bit 7, 6	Plus and minus traversing command
Signal state 1 or edge change 0 → 1	A traverse movement of the axis is to be executed in one or the other direction. Depending on the mode selected, the command is triggered in different ways: • JOG mode: With the plus or minus traverse key. • REF mode: With the traverse key that takes the axis to the reference point. • AUT/MDA mode: A program block containing a coordinate value for the axis in question is executed.
Signal state 0 or edge change 1 → 0	A traversing command in the relevant axis direction has not been given or a traverse movement has been completed. • JOG mode: The traversing command is reset depending on the current setting "jog or continuous mode" (DB31, ... DBX4.7 and/or DBX4.6). While traversing with the handwheel. • REF mode: When the reference point is reached. • AUT/MDA mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. DB21, … DBX25.7 (axes disable) is active.
Application example(s)	To release clamping of axes with clamping (e.g. on a rotary table). **Note:** If the clamping is not released until the traversing command is given, these axes cannot be operated under continuous path control!
Corresponding to ...	DB31, ... DBX4.7 and/or DBX4.6 (traversing key plus and traversing key minus) DB31, ... DBX64.5 and/or DBX.4 (Traversing request plus and minus)

DB31, ... DBB65 Bit 0-6	Active machine function INC1, ..., continuous	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The PLC interface receives a signal stating which JOG mode machine function is active for the machine axes. The reaction to actuation of the traverse key or rotation of the handwheel varies, depending on which machine function is active.	
Signal state 0 or edge change 1 → 0	The machine function in question is not active.	
Corresponding to ...	DB31, ... DBB5 (machine function INC1, ..., continuous)	

DB31, ... DBX67.0	Invert handwheel direction of rotation active (machine axes)	
Edge evaluation: No	Signal(s) updated: Cyclic	
Description	For a handwheel, which is assigned to a machine axis, this signal indicates whether the direction of rotation was inverted:	
	Signal = 1	The direction of rotation of the handwheel is inverted.
	Signal = 0	The direction of rotation of the handwheel is not inverted.
Corresponding to ...	DB31, ... DBX7.0 (invert handwheel direction of rotation for machine axes)	

DB31, ... DBB75 Bit 0-2	JOG - Approaching fixed point active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Message to the PLC that the function "Approaching fixed point in JOG" is effective. The selected machine axis can be traversed to the specified fixed point binary-coded via Bit 0-2 with the traverse keys or the handwheel.
Signal state 0 or edge change 1 → 0	"Approaching fixed point in JOG" is not active
Corresponding to ...	DB31, ... DBX13.0-2 (JOG - Approach fixed point) DB31, ... DBX75.3-5 (JOG - Approach fixed point)

DB31, ... DBB75 Bit 3-5	JOG - Approaching fixed point reached
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Message to PLC that the selected axis has reached the approaching fixed point with "exact stop fine" by virtue of the traversing motion in JOG. This display signal is also signaled if the axis reaches the fixed point position in the machine coordinates system via other methods e.g. NC program, FC18 (for 840D sl) or synchronized action on the setpoint side and comes to a standstill on the actual value side within the "Exact stop fine" tolerance window (MD36010 $MA_STOP_LIMIT_FINE).
Signal state 0 or edge change 1 → 0	The axis has not yet reached the approaching fixed point.
Corresponding to ...	DB31, ... DBX13.0-2 (JOG - Approach fixed point) DB31, ... DBX75.0-2 (JOG - Approach fixed point)

18.4 Compensations (K3)

No signal descriptions required.

18.5 Mode Groups, Channels, Axis Replacement (K5)

18.5.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBB8	Axis/spindle replacement	
Edge evaluation: Yes	Signal(s) updated: Cyclic	
Signal state 1 or edge change 0 → 1	The current axis type and currently active channel for this axis must be specified. With axis replacement by the PLC, the bit meanings of the signal to axis/spindle DB31, ... DBB8:	
	Bit 0:	Assign A NC axis/spindle channel
	Bit 1:	B ...
	Bit 2:	C ....
	Bit 3:	Assign D NC axis/spindle channel
	Bit 4:	Activation, assignment by means of a positive edge
	Bit 5:	-
	Bit 6:	-
	Bit 7:	Request PLC axis/spindle
Signal state 0 or edge change 1 → 0		
Corresponding to	DB31, ... DBB68 (Axis/spindle replacement) MD20070 $MC_AXCONF_ASSIGN_MASTER_USED (Machine axis number valid in channel) MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN (Initial setting of channel for axis replacement)	
Special cases, errors, ...		

18.5.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBB68	Axis/spindle replacement	
Edge evaluation: Yes	Signal(s) updated: Cyclic	
Signal state 1 or edge change 0 → 1	The current axis type and currently active channel for this axis is displayed. With axis replacement by the PLC, the bit meanings of the signal from axis/spindle DB31, ... DBB68:	
	Bit 0:	A NC axis/spindle in channel
	Bit 1:	B ...
	Bit 2:	C ....
	Bit 3:	D NC axis/spindle in channel
	Bit 4:	New type requested from PLC
	Bit 5:	Axis replacement possible
	Bit 6:	neutral axis/spindle as well as command/oscillation axes
	Bit 7:	PLC axis/spindle
Signal state 0 or edge change 1 → 0		
Corresponding to	DB31, ... DBB8 (Axis/spindle replacement) MD20070 $MC_AXCONF_ASSIGN_MASTER_USED (Machine axis number valid in channel) MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN (Initial setting of channel for axis replacement)	
Special cases, errors, ...		

18.6 Kinematic Transformation (M1)

18.6.1 Signals from channel (DB21, ...)

DB21, ... DBX33.6	Transformation active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The NC command `TRANSMIT`, `TRACYL`, `TRAANG` or `TRAORI` is programmed in the part program. The corresponding block has been processed by the NC and a transformation is now active.
Signal state 0 or edge change 1 → 0	No transformation active.
References	Programming Guide Advanced Function Description, Special Functions; 3-Axis to 5-Axis Transformation (F2)

18.7 Measurement (M5)

18.7.1 Signals from NC (DB10)

DB10 DBX107.0 and DBX107.1	**Probe actuated**	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Probe 1 or 2 is actuated.	
Signal state 0 or edge change 1 → 0	Probe 1 or 2 is not actuated.	
References	Equipment Manual NCU	

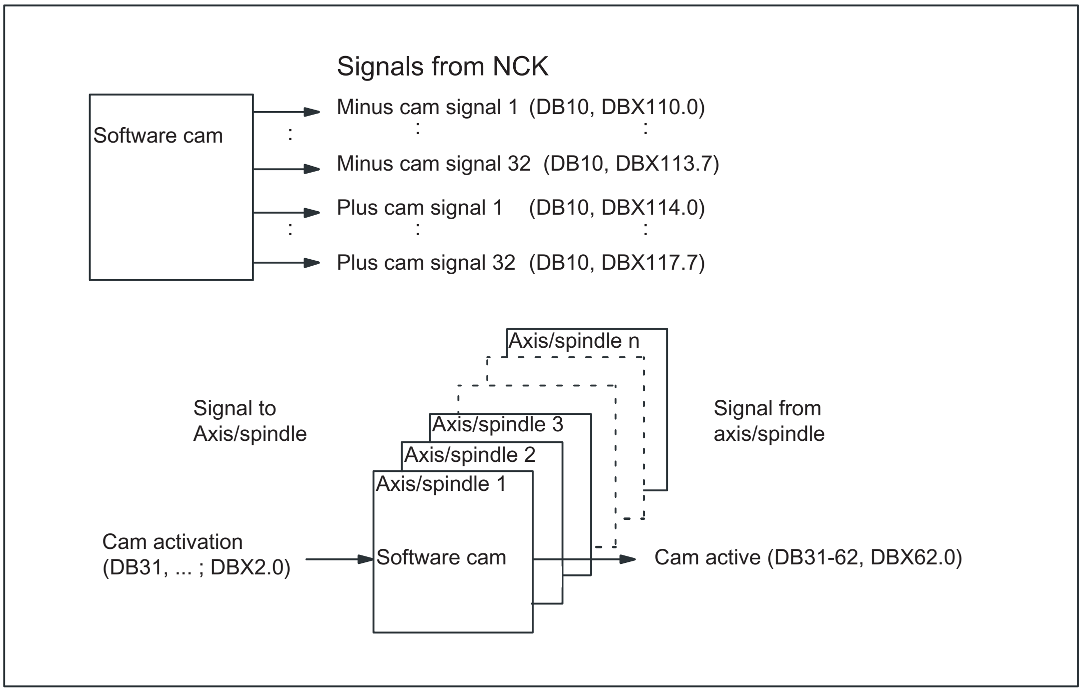
18.7.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX62.3	**Measuring status**	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The "Measuring" function is active. This signal is used during measuring and displays the current measuring status of the axis.	
Signal state 0 or edge change 1 → 0	The "Measuring" function is not active.	

18.8 Software cams, position switching signals (N3)

18.8.1 Signal overview

PLC interface signals for "Software cams, position switching signals"



18.8.2 Signals from NC (DB10)

DB10 DBX110.0-113.7	**Minus cam signal 1-32**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The switching edge of the minus cam signal 1-32 is generated as a function of the traversing direction of the (rotary) axis and transferred to the PLC interface in the IPO cycle. **Linear axis:** The minus cam signal switches from 0 to 1 if the axis overtravels the minus cam in the negative axis direction. **Modulo rotary axis:** The minus cam signal changes level in response to every positive edge of the plus cam signal.
Signal state 0 or edge change 1 → 0	**Linear axis:** The minus cam signal switches from 1 to 0 when the axis traverses the minus cam in the positive axis direction. **Modulo rotary axis:** The minus cam signal changes level in response to every positive edge of the plus cam signal.

DB10 DBX114.0-117.7	**Plus cam signal 1-32**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The switching edge of the plus cam signal 1-32 is generated as a function of the traversing direction of the (rotary) axis and transferred to the PLC interface in the IPO cycle. **Linear axis:** The plus cam signal switches from 0 to 1 when the axis traverses the plus cam in the positive direction. **Modulo rotary axis:** The plus cam signal switches from 0 to 1 when the minus cam is overtraveled in the positive axis direction. The described response of the plus cam applies under the condition: plus cam - minus cam < 180 degrees If this condition is not fulfilled or if the minus cam is set to a greater value than the plus cam, then the response of the plus cam signal is inverted. The response of the minus cam signal remains unchanged.
Signal state 0 or edge change 1 → 0	**Linear axis:** The plus cam signal switches from 1 to 0 if the axis overtravels the plus cam in the negative direction. **Modulo rotary axis:** The plus cam signal switches from 1 back to 0 if the plus cam is overtraveled in the positive axis direction. The described response of the plus cam applies under the condition: plus cam - minus cam < 180 degrees If this condition is not fulfilled or if the minus cam is set to a greater value than the plus cam, then the response of the plus cam signal is inverted. The response of the minus cam signal remains unchanged.

18.8.3 Signals to axis/spindle (DB31, ...)

DB31, ... DBX2.0	Cam activation	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Output of the minus and plus cam signals of an axis to the general PLC interface is activated. The activation takes effect immediately after processing of IS "Cam activation".	
Signal state 0 or edge change 1 → 0	The minus and plus cam signals of an axis are **not** output to the general PLC interface.	
Corresponding to	DB10 DBX110.0 - 113.7 (minus cam signal 1-32) DB10 DBX114.0 - 117.7 (plus cam signals 1-32)	

18.8.4 Signals from axis/spindle (DB31, ...)

DB31, ... DBX62.0	Cams active	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	All cams of the axis selected via NC/PLC interface signal: DB31, ... DBX2.0 (Cam activation) have been activated successfully.	
Signal state 0 or edge change 1 → 0	The cams of the axis are **not** activated.	
Corresponding to	DB31, ... DBX2.0 (Cam activation) DB10 DBX110.0 - 113.7 (minus cam signal 1-32) DB10 DBX114.0 - 117.7 (plus cam signals 1-32)	

18.9 Punching and Nibbling (N4)

18.9.1 Signal overview

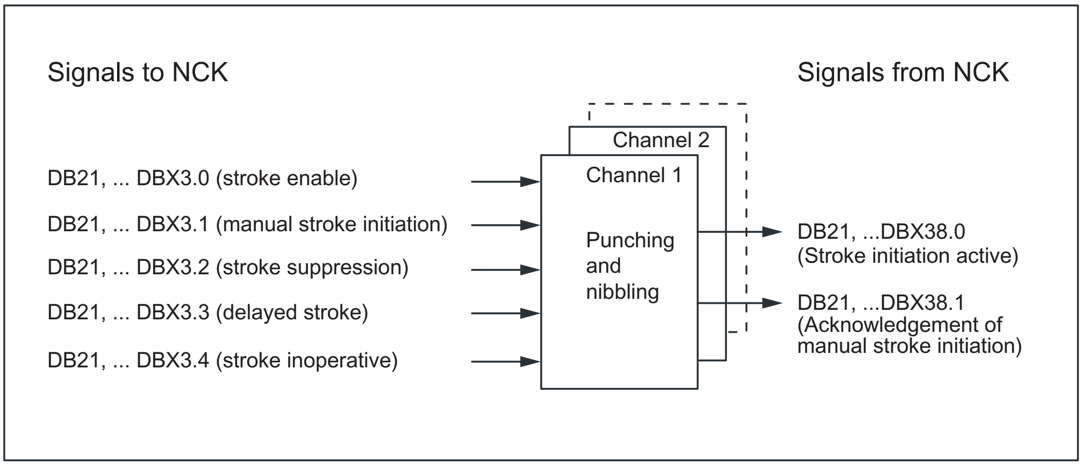


Figure 18-1 PLC interface signals for "Punching and nibbling"

18.9.2 Signals to channel (DB21, ...)

DB21, ... DBX3.0		No stroke enable
Edge evaluation:		Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal releases the punching strokes via the PLC. 1 signal: The stroke is locked, the NC may not trigger a punching stroke.	
Signal state 0 or edge change 1 → 0	0 signal: Punching stroke is available. As long as release is not set, the NC may perform a punching stroke	

DB21, ... DBX3.1		Manual stroke initiation
Edge evaluation:		Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal enables the triggering of a single stroke in manual mode. 1 signal: Manual stroke is performed.	
Signal state 0 or edge change 1 → 0	0 signal: No effect.	

DB21, ... DBX3.2	Stroke suppression
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal simply prevents execution of the stroke. The machine traverses anyway. The automatic path segmentation remains active if it is already activated. Only the signal "Stroke initiation" is suppressed. The machine traverses in "stop and go" mode. The step length is defined via the path segmentation. 1 signal: Stroke suppression is active.
Signal state 0 or edge change 1 → 0	0 signal: Stroke suppression is not active.

DB21, ... DBX3.3	Delayed stroke
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	A "Delayed stroke" can be activated via this signal. This corresponds in function to the programming of `PDELAYON`. Other PLC signals not corresponding to the standard are not evaluated in the NCK. With the exception of the manual stroke initiation, the evaluation of signals is limited to `PON` active. 1 signal: Delayed stroke is active.
Signal state 0 or edge change 1 → 0	0 signal: Delayed stroke is not active.

DB21, ... DBX3.4	Stroke inoperative
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	The NC reacts to this signal by initiating an immediate movement stop. An alarm is output if any other movement or action needs to be interrupted as a result of this signal. In physical terms, the signal is identical to the signal "Stroke active" for the CNC, i.e. the system is wired in such a way that the two signals are taken to the same NC input via an AND gate. 1 signal: Stroke inoperative (corresponds to the signal "stroke enable").
Signal state 0 or edge change 1 → 0	0 signal: Stroke operative (corresponds to the signal "stroke enable").

DB21, ... DBX3.5		Manual stroke initiation
Edge evaluation:		Signal(s) updated:
Signal state 1 or edge change 0 → 1	The signal "manual stroke initiation" allows the operator to initiate a punching process, even when the parts program is not being processed. Thus the initiation of the punching process is controlled from the PLC. Successful stroke initiation is indicated to the PLC by the NCK-PLC interface signal: DB21, ... DBX38.1 (Manual stroke initiation acknowledgement) . 1 signal: Manual stroke initiation is active.	
Signal state 0 or edge change 1 → 0	0 signal: Manual stroke initiation is not active.	

18.9.3 Signals from channel (DB21, ...)

DB21, ... DBX38.0		Stroke initiation active
Edge evaluation:		Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal displays whether the stroke initiation is active. 1 signal: Stroke initiation is active.	
Signal state 0 or edge change 1 → 0	0 signal: Stroke initiation is not active.	

DB21, ... DBX38.1		Acknowledgement of manual stroke initiation
Edge evaluation:		Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal displays whether a manual stroke has been initiated. 1 signal: Manual stroke has been performed.	
Signal state 0 or edge change 1 → 0	0 signal: Manual stroke has not been performed.	

18.10 Positioning axes (P2)

The following signals or commands on the NCK-HMI-PLC interface are only of significance for the positioning axis:

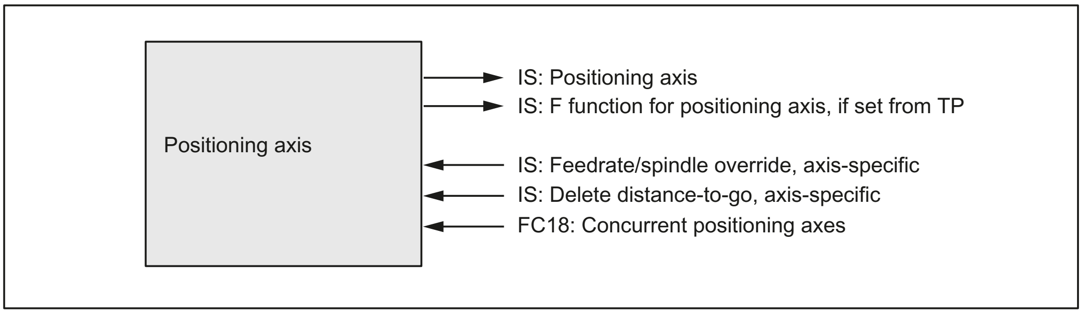


Figure 18-2 Signal modification by the PLC

18.10.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBB0	**Feedrate override / spindle override axis-specific**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Positioning axes have their own axis-specific feedrate override value. This feedrate override is evaluated in the same way as the channel-specific feedrate override.
Signal irrelevant for ...	NST DB31, ... DBX74.5 ("Positioning axis") = ZERO
References	Evaluation see: DB21, ... DBB4 (feedrate override); channel-specific

DB31, ... DBX2.2	**Delete distance-to-go, axis-specific**
Edge evaluation: Yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis-specific distance-to-go of the positioning axis is canceled. The positioning axis is decelerated and the following error is eliminated. The programmed end position is deemed to have been reached. The path axes are not influenced by the axis-specific "delete distance-to-go" interface signal. The channel-specific "delete distance-to-go" interface signal is used for this purpose.
Special cases, errors, ...	If the axis-specific "delete distance-to-go" interface signal is enabled, even if no positioning axes have been programmed in this block, the NCK does not respond.
Corresponding to ...	DB21, ... DBX6.2 (delete distance-to-go); channel-specific for path axes

DB31, ... DBX28.1	Reset	
Edge evaluation: Yes		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Reset request to the NCK for the PLC-controlled axis/spindle. Feedback signal from the NCK to the PLC: DB31 ... DBX63.1 = 1 (reset executed) DB31 ... DBX63.2 = 1 (axis stop active)	
Special cases, errors, ...	Boundary condition: • The axis/spindle must be currently controlled by the PLC.	
Corresponding to ...	DB31 ... DBX63.1 (reset executed) DB31, ... DBX63.2 (axis stop active) System variable: $AA_SNGLAX_STAT OPI variables: aaSnglAxStat	

DB31, ... DBX28.2	Continue	
Edge evaluation: Yes		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Request to continue interrupted traversing motion for a PLC-controlled axis/spindle. The request can be interrupted with DB31 … DBX63.2 ("axis stop active").	
Special cases, errors, ...	Boundary condition: • The axis/spindle must be currently controlled by the PLC. • The signal is ignored for following error situations: – The axis/spindle is not controlled by the PLC. – The axis/spindle is not in the stopped state. – The axis/spindle must not resume traversing because an alarm is present.	
Corresponding to ...	DB31, … DBX28.1 (reset) DB31, ... DBX60.6 (exact stop coarse) DB31, ... DBX60.7 (exact stop fine) DB31 ... DBX63.2 (axis stop active) DB31, ... DBX64.6 (traversing command minus) DB31, ... DBX64.7 (traversing command plus) System variable: $AA_SNGLAX_STAT OPI variables: aaSnglAxStat	

DB31, ... DBX61.1	Axial alarm	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Effects: • The axis/spindle is stopped by the NCK via a braking ramp. • OPI variables: aaSnglAxStat = 5 (alarm) • $AA_SNGLAX_STAT = 5 (axial alarm is present) • DB31 ... DBX61.1 = 1 (axial alarm)	

DB31, ... DBX63.0	Reset executed	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The following state is present after the reset: • The machine data of the axis/spindle is reloaded • DB31 ... DBX63.0 == 1 (reset executed) • DB31 ... DBX63.2 == 0 (axis stop active) • System variable $AA_SNGLAX_STAT == 1 • OPI variables: aaSnglAxStat == 1	
Corresponding to ...	DB31, … DBX28.1 (reset)	

DB31, ... DBX63.1	PLC-controlled axis	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Confirmation of the NC to the PLC that the axis is now controlled by the PLC.	
Corresponding to ...	DB31 ... DBX28.7 (PLC controls the axis) System variable: $AA_SNGLAX_STAT	

DB31, … DBX63.2	Axis stop active	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Signal from the NC to the PLC that the axis will be stopped.	

DB31, … DBX63.2	Axis stop active
Signal state 0 or edge change 1 → 0	Confirmation from the NC to the PLC that the axis has been stopped. System variable: $AA_SNGLAX_STAT = 3 (single axis is interrupted)
Corresponding to ...	DB31, ... DBX60.6 (exact stop coarse) DB31, ... DBX60.7 (exact stop fine) DB31 ... DBX63.2 (axis stop active) DB31, ... DBX64.6 (traversing command minus) DB31, ... DBX64.7 (traversing command plus) System variable: $AA_SNGLAX_STAT

DB31, ... DBX76.5	Positioning axis
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Confirmation from the NC to the PLC that the axis is a positioning axis.

DB31, ... DBD78	F function (feedrate) for positioning axis
Edge evaluation: No	Signal(s) updated: when change
Function	The axial feedrate programmed for the positioning axis. The value specified by FC18 for 840D sl is not output.
Signal irrelevant for ...	DB31, ... DBX76.5 == 0 (axis is not a positioning axis)
Special cases, errors, ...	If the positioning axis is traversed with the feedrate from the machine data, the NC does not output an F function (feed) to the PLC: MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)
Corresponding to ...	DB31, ... DBX76.5 (positioning axis) MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

18.10.2 Function call - only 840D sl

FC18

For SINUMERIK 840D sl, concurrent positioning axes can be started from the PLC using FC18 (Function Call 18) of the PLC. The following parameters are passed to the function call:

- Axis name/axis number
- End position
- Feedrate

  (for feedrate = 0, the feedrate is taken from MD32060 $MA_POS_AX_VELO)

  The F value of FC18 is **not**transferred to the axis-specific IS DB31, ...DBB78-81 ("F function (feedrate) for positioning axis")
- Absolute coordinates (G90), incremental coordinates (G91), absolute coordinates along the shortest path for rotary axes (rotary axis name = DC(value))

Since each axis is assigned to exactly one channel, the control can select the correct channel from the axis name/axis number and start the concurrent positioning axis on this channel.

Reference:
Function Manual Basic Functions; PLC Basic Program for SINUMERIK 840D sl (P3)

18.11 Oscillation (P5)

18.11.1 Signals to axis/spindle (DB31, ...)

VDI input signals

The PLC user program uses the following signals to control the oscillation process.

DB31, ... DBX28.0	External oscillation reversal
Edge evaluation: Yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Brake oscillation motion and move oscillation axis in the opposite direction.
Signal state 0 or edge change 1 → 0	Continue oscillation without interruption

DB31, ... DBX28.3	Set reversal point
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Reversal point 2
Signal state 0 or edge change 1 → 0	Reversal point 1

DB31, … DBX28.4	Alter reversal point
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The selected reversal point can be altered by manual traverse. In conjunction with DB31, ...DBX28.0: The position at which axis is braked after external oscillation reversal must be accepted as new reversal point.
Signal state 0 or edge change 1 → 0	The selected reversal point cannot be altered by manual traverse. In conjunction with DB31, ...DBX28.0: No change to reversal point
Corresponding to	DBX28.3

DB31, ... DBX28.5	Stop at next reversal point
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The oscillation movement is interrupted at the next reversal point.
Signal state 0 or edge change 1 → 0	The oscillation movement continues after the next reversal point.
Corresponding to	DBX28.6, DBX28.7

DB31, ... DBX28.6	Stop along braking ramp
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is decelerated along a ramp, the oscillation movement is interrupted.
Signal state 0 or edge change 1 → 0	The oscillation movement continues without interruption.
Corresponding to	DBX28.5, DBX28.7

DB31, ... DBX28.7	PLC controls axis
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Axis is controlled by the PLC. The reaction to interface signals is controlled by the PLC by means of the 2 stop bits, other signals with deceleration action are ignored.
Signal state 0 or edge change 1 → 0	Axis is not controlled by the PLC.
Corresponding to	DBX28.5, DBX28.6

18.11.2 Signals from axis/spindle (DB31, ...)

VDI output signals

The NCK makes the following signals available to the PLC user program.

DB31, ... DBX100.2	Oscillation reversal active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The deceleration period after external oscillation reversal (DB31, ...DBX28.0) is active
Signal state 0 or edge change 1 → 0	No deceleration after external oscillation reversal is active

DB31, ... DBX100.3	Oscillation cannot start
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The oscillation axis cannot be started owing to incorrect programming. This status can occur even when axis has already been traversed.
Signal state 0 or edge change 1 → 0	The oscillation movement can be started.

DB31, ... DBX100.4	Error during oscillation movement
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	The oscillation movement has been aborted.
Signal state 0 or edge change 1 → 0	The oscillation movement is being executed correctly.

DB31, … DBX100.5	Sparking-out active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is executing sparking-out strokes.
Signal state 0 or edge change 1 → 0	The axis is not currently executing sparking-out strokes.
Corresponding to	DBX100.7

DB31, ... DBX100.6	Oscillation movement active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is executing an oscillation movement between 2 reversal points.
Signal state 0 or edge change 1 → 0	The axis is not currently oscillating.
Signal irrelevant for	DBX100.7 = 0
Corresponding to	DBX100.7

DB31, ... DBX100.7	Oscillation active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is currently being traversed as an oscillation axis.
Signal state 0 or edge change 1 → 0	The axis is a positioning axis.
Corresponding to	DBX100.5, DBX100.6

DB31, ... DBX104.0 - 7	Active infeed axes
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis sending the signal is currently the oscillation axis and is indicating its active infeed axes in this field (104.0 axis 1 is infeed axis, 104.1 axis 2 is infeed axis, etc.).
Signal state 0 or edge change 1 → 0	The associated axis is not an infeed axis.
Corresponding to	DBX100.7

18.12 Rotary axes (R2)

18.12.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX12.4	**Traversing range limitation for modulo rotary axes**	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Activate traversing range limitation for modulo rotary axes (software end switches, work field limitations).	
Signal state 0 or edge change 1 → 0	Deactivate traversing range limitation for modulo rotary axes.	
Signal irrelevant for ...	Linear axes / rotary axes without modulo functionality.	
Application example(s)	Built-on rotary axis with monitoring	

18.12.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX74.4	**Monitoring status with modulo rotary axes**	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Traversing range limitation for modulo rotary axes active (software end switches, work field limitations).	
Signal state 0 or edge change 1 → 0	Traversing range limitation for modulo rotary axes not active.	
Signal irrelevant for ...	Linear axes / rotary axes without modulo functionality.	
Application example(s)	Built-on rotary axis with monitoring	

18.13 Synchronous Spindles (S3)

18.13.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX31.5	Disable synchronization
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The synchronization motion for the following spindle is not disabled from the PLC. The position offset is not suppressed and applied as in earlier versions.
Signal state 0 or edge change 1 → 0	The synchronization motion for the following spindle is disabled from the PLC. A synchronization motion specified via offset programming is suppressed for the following spindle. The following spindle does not execute any additional movement.
Corresponding to	DB31, ... DBX98.1 (Synchronism coarse) DB31, ... DBX98.0 (Synchronism fine)

18.13.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX84.4	Synchronous mode
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The spindle is operating in "Synchronous operation" mode. The following spindle thus follows the movements of the leading spindle in accordance with the transmission ratio. The monitoring functions for coarse and fine synchronism are implemented in synchronous operation. Note: The signal is set only for the machine axis which is acting as following spindle (IS "FS active" = 1)
Signal state 0 or edge change 1 → 0	The spindle is not operated as the following spindle in "synchronous mode". When the coupling is deactivated (deselection of synchronous operation), the following spindle is switched to "open-loop control mode".
Corresponding to	DB31, ... DBX98.0 (Synchronism fine) DB31, ... DBX98.1 (Synchronism coarse) DB31, ... DBX99.1 (FS active)

DB31, ... DBX98.0	Fine synchronism
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The positional deviation or velocity difference between the following spindle and its leading spindle is within the "Fine synchronism" tolerance band.
Signal state 0 or edge change 1 → 0	The positional deviation or velocity difference between the following spindle and its leading spindle is not within the "Fine synchronism" tolerance band. Note: The signal is relevant only for the following spindle in synchronous operation.
Application example	Clamping of workpiece in following spindle on transfer from the leading spindle: Clamping of the workpiece is not initiated by the PLC user program until the spindles are sufficiently synchronized.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) MD37210 $MA_COUPLE_POS_TOL_FINE (threshold value for "fine synchronism") MD37230 $MA_COUPLE_VELO_TOL_FINE ("fine" speed tolerance)

DB31, ... DBX98.1	Coarse synchronism
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The positional deviation or velocity difference between the following spindle and its leading spindle is within the "Coarse synchronism" tolerance band. Note: The signal is relevant only for the following spindle in synchronous operation.
Signal state 0 or edge change 1 → 0	The positional deviation or velocity difference between the following spindle and its leading spindle is not within the "Coarse synchronism" tolerance band.
Application example	Clamping of workpiece in following spindle on transfer from the leading spindle: Clamping of the workpiece is not initiated by the PLC user program until the spindles are sufficiently synchronized.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) MD37200 $MA_COUPLE_POS_TOL_COARSE (threshold value for "coarse synchronism") MD37220 $MA_COUPLE_VELO_TOL_COARSE ("coarse" speed tolerance)

DB31, ... DBX98.2	Actual value coupling
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The actual-value coupling is active as the coupling type between the leading and following spindles (see MD21310). Note: The signal is relevant only for the active following spindle in synchronous operation.
Signal state 0 or edge change 1 → 0	The setpoint coupling is active as the coupling type between the leading and following spindles (see MD21310).

DB31, ... DBX98.2	Actual value coupling
Special cases, errors, ...	In the case of faults/disturbances on the following spindle which result in cancellation of the FS "servo enable", the coupling relationship between the FS and LS is reversed and switched over to an actual-value coupling internally in the control under certain circumstances.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) MD21310 $MC_COUPLING_MODE_1 (coupling type in synchr. spindle oper.)

DB31, ... DBX98.4	Overlaid motion
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The following spindle traverses an additional motional component which is overlaid on the motion from the coupling with the leading spindle. Examples of overlaid movement of FS: - Activation of synchronous operation with defined angular offset between FS and LS - Activation of synchronous operation with LS in rotation - Alteration of transmission ratio when synchronous operation is selected - Input of a new defined angular offset when synchronous operation is selected - Traversal of FS with plus or minus traversing keys or handwheel in JOG when synchronous operation is selected As soon as the FS executes an overlaid movement, IS "Fine synchronism" or IS "Coarse synchronism" (depending on threshold value) may be canceled immediately. Note: The signal is relevant only for the following spindle in synchronous operation.
Signal state 0 or edge change 1 → 0	The following spindle does not traverse any additional motional component or this motion has been terminated.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode)

DB31, ... DBX99.0	LS (leading spindle) active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The machine axis is currently active as the leading spindle. Note: The signal is relevant only in synchronous operation.
Signal state 0 or edge change 1 → 0	The machine axis is not currently active as the leading spindle.
Special cases, errors, ...	In the case of faults/disturbances on the following spindle which result in cancellation of the FS "servo enable", the coupling relationship between the FS and LS is reversed and switched over to an actual-value coupling internally in the control under certain circumstances. In this case, the leading spindle becomes the new, active following spindle (IS "FS active").
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) DB31, ... DBX99.1 (FS active)

DB31, ... DBX99.1	FS (following spindle) active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The machine axis is currently operating as the following spindle. The following spindle thus follows the movements of the leading spindle in synchronous operation in accordance with the transmission ratio. Note: The signal is relevant only in synchronous operation.
Signal state 0 or edge change 1 → 0	The machine axis is not currently operating as the following spindle.
Special cases, errors, ...	In the case of faults/disturbances on the following spindle which result in cancellation of the FS "servo enable", the coupling relationship between the FS and LS is reversed and switched over to an actual-value coupling internally in the control under certain circumstances.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) DB31, ... DBX99.0 (LS active)

18.14 Memory Configuration (S7)

No signal descriptions required.

18.15 Indexing Axes (T1)

18.15.1 Signals from axis/spindle (DB31, ...)

DB31, ... DBX76.6	Indexing axis in position
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The signal is influenced according to the "Exact stop fine": When "Exact stop fine" is achieved, the signal is set. When exiting "Exact stop fine", the signal is reset. • The indexing axis is located on an indexing position. The indexing axis has been positioned with instructions for "Coded Position". **Note:** If the "Exact stop fine" window is reached and the indexing axis is positioned on an indexing position, the signal is enabled regardless of how the indexing position was reached.

DB31, ... DBX76.6	Indexing axis in position
Signal state 0 or edge change 1 → 0	• The axis is not defined as an indexing axis. • The indexing axis is traversing: DB31, ... DBX64.7/64.6 (Travel command+/-) is active. • The indexing axis is located at a position which is not an indexing position. Examples: – In JOG mode after abortion of travel movement, e.g. with RESET – in Automatic mode: indexing axis has, for example, approached a selected position controlled by an AC or DC instruction • The indexing axis has not been positioned with instructions for coded positions (CAC, CACP, CACN, CDC, CIC) in automatic mode. • The "Servo enable" signal for the indexing axis has been canceled: DB31, ... DBX2.1 (Servo enable)
Signal irrelevant for ...	... Axes that are not defined as indexing axes: MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB = 0
Application example(s)	Tool magazine: Activation of a gripper for removing a tool from a magazine is triggered when the indexing axis is in position: DB31, ... DBX76.6 (indexing axis in position) = 1. This must ensured by the PLC user program.
Special cases, errors, ...	**Notes:** The axis positions entered in the indexing position table for the individual divisions can be changed through zero offsets (including DRF). The interface signal: DB31, ... DBX76.6 (indexing axis in position) is then set to 1 when the actual position of the indexing axis matches the value entered in the index table plus the offset. If a DRF is applied to an indexing axis in AUTOMATIC mode, then interface signal "Indexing axis in position" remains active even though the axis is no longer at an indexing position.
Corresponding to	MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB (axis is an indexing axis)

18.16 Tool Change (W3)

No signal descriptions required.

18.17 Grinding-specific tool offset and tool monitoring (W4)

18.17.1 Signals from axis/spindle (DB31, ...)

DB31, ... DBX83.3	Geometry monitoring
Edge evaluation: No	Signal(s) updated: -
Signal state 1 or edge change 0 → 1	Error in grinding wheel geometry. **Note:** There is no further reaction to the response of this monitoring function. Reactions deemed necessary must be programmed by the PLC user.
Signal state 0 or edge change 1 → 0	No error in grinding wheel geometry.
Application example(s)	Grinding-specific tool monitoring

DB31, ... DBX83.6	Speed monitoring
Edge evaluation: No	Signal(s) updated: -
Signal state 1 or edge change 0 → 1	Error in grinding wheel speed. **Note:** No further reaction to this signal state is programmed. Reactions deemed necessary must be programmed by the PLC user.
Signal state 0 or edge change 1 → 0	No error in grinding wheel speed.
Application example(s)	Grinding-specific tool monitoring

DB31, ... DBX84.1	GWPS active
Edge evaluation: No	Signal(s) updated: -
Signal state 1 or edge change 0 → 1	Constant grinding wheel peripheral speed (GWPS) is active. If GWPS is active, then all S value inputs from the PLC are interpreted as the grinding wheel peripheral speed.
Signal state 0 or edge change 1 → 0	Constant grinding wheel peripheral speed (GWPS) is not active.
Application example(s)	GWPS in all operating modes.

Appendix A

A.1 List of abbreviations

A	
O	Output
ADI4	(Analog drive interface for 4 axes)
AC	Adaptive Control
ALM	Active Line Module
ARM	Rotating induction motor
AS	PLC
ASCII	American Standard Code for Information Interchange: American coding standard for the exchange of information
ASIC	Application-Specific Integrated Circuit: User switching circuit
ASUB	Asynchronous subprogram
AUXFU	Auxiliary Function: Auxiliary function
STL	Statement List
UP	User Program

B	
BA	Mode
BAG	Mode group
BCD	Binary Coded Decimals: Decimal numbers encoded in binary code
BERO	Proximity limit switch with feedback oscillator
BI	Binector Input
BICO	Binector Connector
BIN	BINary files: Binary files
BIOS	Basic Input Output System
BCS	Basic Coordinate System
BO	Binector Output
OPI	Operator Panel Interface

C	
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CC	Compile Cycle: Compile cycles

C	
CI	Connector Input
CF Card	Compact Flash Card
CNC	Computerized Numerical Control: Computer-Supported Numerical Control
CO	Connector Output
CoL	Certificate of License
COM	Communication
CPA	Compiler Projecting Data: Configuring data of the compiler
CRT	Cathode Ray Tube picture tube
CSB	Central Service Board: PLC module
CU	Control Unit
CP	Communication Processor
CPU	Central Processing Unit: Central processing unit
CR	Carriage Return
CTS	Clear To Send: Ready to send signal for serial data interfaces
CUTCOM	Cutter radius Compensation: Tool radius compensation

D	
DAC	Digital-to-Analog Converter
DB	Data Block (PLC)
DBB	Data Block Byte (PLC)
DBD	Data Block Double word (PLC)
DBW	Data Block Word (PLC)
DBX	Data block bit (PLC)
DDE	Dynamic Data Exchange
DIN	Deutsche Industrie Norm
DIO	Data Input/Output: Data transfer display
DIR	Directory: Directory
DLL	Dynamic Link Library
DO	Drive Object
DPM	Dual Port Memory
DPR	Dual Port RAM
DRAM	Dynamic memory (non-buffered)
DRF	Differential Resolver Function: Differential revolver function (handwheel)
DRIVE-CLiQ	Drive Component Link with IQ
DRY	Dry Run: Dry run feedrate
DSB	Decoding Single Block: Decoding single block
DSC	Dynamic Servo Control / Dynamic Stiffness Control
DW	Data Word
DWORD	Double Word (currently 32 bits)

I	
I	Input
I/O	Input/Output
ENC	Encoder: Actual value encoder
EFP	Compact I/O module (PLC I/O module)
ESD	Electrostatic Sensitive Devices
EMC	ElectroMagnetic Compatibility
EN	European standard
EnDat	Encoder interface
EPROM	Erasable Programmable Read Only Memory: Erasable, electrically programmable read-only memory
ePS Network Services	Services for Internet-based remote machine maintenance
EQN	Designation for an absolute encoder with 2048 sine signals per revolution
ES	Engineering System
ESR	Extended Stop and Retract
ETC	ETC key ">"; softkey bar extension in the same menu

F	
FB	Function Block (PLC)
FC	Function Call: Function Block (PLC)
FEPROM	Flash EPROM: Read and write memory
FIFO	First In First Out: Memory that works without address specification and whose data is read in the same order in which they was stored
FIPO	Fine interpolator
FPU	Floating Point Unit: Floating Point Unit
CRC	Cutter Radius Compensation
FST	Feed Stop: Feedrate stop
FBD	Function Block Diagram (PLC programming method)
FW	Firmware

G	
GC	Global Control (PROFIBUS: Broadcast telegram)
GEO	Geometry, e.g. geometry axis
GIA	Gear Interpolation dAta: Gear interpolation data
GND	Signal Ground
GP	Basic program (PLC)
GS	Gear Stage
GSD	Device master file for describing a PROFIBUS slave

G	
GSDML	Generic Station Description Markup Language: XML-based description language for creating a GSD file
GUD	Global User Data: Global user data

H	
HEX	Abbreviation for hexadecimal number
AuxF	Auxiliary Function
HLA	Hydraulic linear drive
HMI	Human Machine Interface: SINUMERIK user interface
MSD	Main Spindle Drive
HW	Hardware

I	
IBN	Commissioning
ICA	Interpolatory compensation
IM	Interface Module Interconnection module
IMR	Interface Module Receive: Interface module for receiving data
IMS	Interface Module Send: Interface module for sending data
INC	Increment: Increment
INI	Initializing Data: Initializing data
IPO	Interpolator
ISA	Industry Standard Architecture
ISO	International Standardization Organization

J	
JOG	Jogging: Setup mode

K	
K_V	Gain factor of control loop
K_p	Proportional gain
$K_Ü$	Transformation ratio
LAD	Ladder Diagram (PLC programming method)

L	
LAI	Logic Machine Axis Image: Logical machine axes image
LAN	Local Area Network
LCD	Liquid Crystal Display: Liquid crystal display
LED	Light Emitting Diode: Light-emitting diode
LF	Line Feed
PMS	Position Measuring System
LR	Position controller
LSB	Least Significant Bit Least significant bit
LUD	Local User Data: User data (local)

M	
MAC	Media Access Control
MAIN	Main program: Main program (OB1, PLC)
MB	Megabyte
MCI	Motion Control Interface
MCIS	Motion Control Information System
MCP	Machine Control Panel: Machine control panel
MD	Machine Data
MDA	Manual Data Automatic: Manual input
MSGW	Message Word
MCS	Machine Coordinate System
MLFB	Machine-readable product code
MM	Motor Module
MPF	Main Program File: Main program (NC)
MCP	Machine Control Panel

N	
NC	Numerical Control: Numerical Control
NCK	Numerical Control Kernel: NC kernel with block preparation, traversing range, etc.
NCU	Numerical Control Unit: NCK hardware unit
NRK	Name for the operating system of the NCK
IS	Interface Signal
NURBS	Non-Uniform Rational B-Spline
ZO	Zero Offset
NX	Numerical Extension: Axis expansion board

O	
OB	Organization block in the PLC
OEM	Original Equipment Manufacturer
OP	Operator Panel: Operating equipment
OPI	Operator Panel Interface: Interface for connection to the operator panel
OPT	Options: Options
OLP	Optical Link Plug: Fiber optic bus connector
OSI	Open Systems Interconnection: Standard for computer communications

P	
PIQ	Process Image Output
PII	Process Image Input
PC	Personal Computer
PCIN	Name of the SW for data exchange with the controller
PCMCIA	Personal Computer Memory Card International Association: Plug-in memory card standardization
PCU	PC Unit: PC box (computer unit)
PG	Programming device
PKE	Parameter identification: Part of a PIV
PIV	Parameter identification: Value (parameterizing part of a PPO)
PLC	Programmable Logic Control: Adaptation control
PN	PROFINET
PNO	PROFIBUS user organization
PO	POWER ON
POU	Program Organization Unit
POS	Position/positioning
POSMO A	Positioning Motor Actuator: Positioning motor
POSMO CA	Positioning Motor Compact AC: Complete drive unit with integrated power and control module as well as positioning unit and program memory; AC infeed
POSMO CD	Positioning Motor Compact DC: Like CA but with DC infeed
POSMO SI	Positioning Motor Servo Integrated: Positioning motor, DC infeed
PPO	Parameter Process data Object: Cyclic data telegram for PROFIBUS DP transmission and "Variable speed drives" profile
PPU	Panel Processing Unit (central hardware for a panel-based CNC, e.g SINUMERIK 828D)
PROFIBUS	Process Field Bus: Serial data bus
PRT	Program Test
PSW	Program control word
PTP	Point-To-Point Point-To-Point
PUD	Program global User Data: Program-global user variables
PZD	Process data: Process data part of a PPO

Q	
QEC	Quadrant Error Compensation

R	
RAM	Random Access Memory: Read/write memory
REF	REFerence point approach function
REPOS	REPOSition function
RISC	Reduced Instruction Set Computer: Type of processor with small instruction set and ability to process instructions at high speed
ROV	Rapid Override: Input correction
RP	R Parameter, arithmetic parameter, predefined user variable
RPA	R Parameter Active: Memory area on the NCK for R parameter numbers
RPY	Roll Pitch Yaw: Rotation type of a coordinate system
RTLI	Rapid Traverse Linear Interpolation: Linear interpolation during rapid traverse motion
RTS	Request To Send: Control signal of serial data interfaces
RTCP	Real Time Control Protocol

S	
SA	Synchronized Action
SBC	Safe Brake Control: Safe Brake Control
SBL	Single Block: Single block
SBR	Subroutine: Subroutine (PLC)
SD	Setting Data
SDB	System Data Block
SEA	Setting Data Active: Identifier (file type) for setting data
SERUPRO	SEarch RUn by PROgram test: Search run by program test
SFB	System Function Block
SFC	System Function Call
SGE	Safety-related input
SGA	Safety-related output
SH	Safe standstill
SIM	Single Inline Module
SK	Softkey
SKP	Skip: Function for skipping a part program block
SLM	Synchronous Linear Motor
SM	Stepper Motor
SMC	Sensor Module Cabinet Mounted
SME	Sensor Module Externally Mounted
SMI	Sensor Module Integrated

S	
SPF	Sub Routine File: Subprogram (NC)
PLC	Programmable Logic Controller
SRAM	Static RAM (non-volatile)
TNRC	Tool Nose Radius Compensation
SRM	Synchronous Rotary Motor
LEC	Leadscrew Error Compensation
SSI	Serial Synchronous Interface: Synchronous serial interface
SSL	Block search
STW	Control word
GWPS	Grinding Wheel Peripheral Speed
SW	Software
SYF	System Files: System files
SYNACT	SYNchronized ACTion: Synchronized Action

T	
TB	Terminal Board (SINAMICS)
TCP	Tool Center Point: Tool tip
TCP/IP	Transport Control Protocol / Internet Protocol
TCU	Thin Client Unit
TEA	Testing Data Active: Identifier for machine data
TIA	Totally Integrated Automation
TM	Terminal Module (SINAMICS)
TO	Tool Offset: Tool offset
TOA	Tool Offset Active: Identifier (file type) for tool offsets
TRANSMIT	Transform Milling Into Turning: Coordination transformation for milling operations on a lathe
TTL	Transistor-Transistor Logic (interface type)
TZ	Technology cycle

U	
UFR	User Frame: Zero Offset
SR	Subprogram
USB	Universal Serial Bus
UPS	Uninterruptible Power Supply

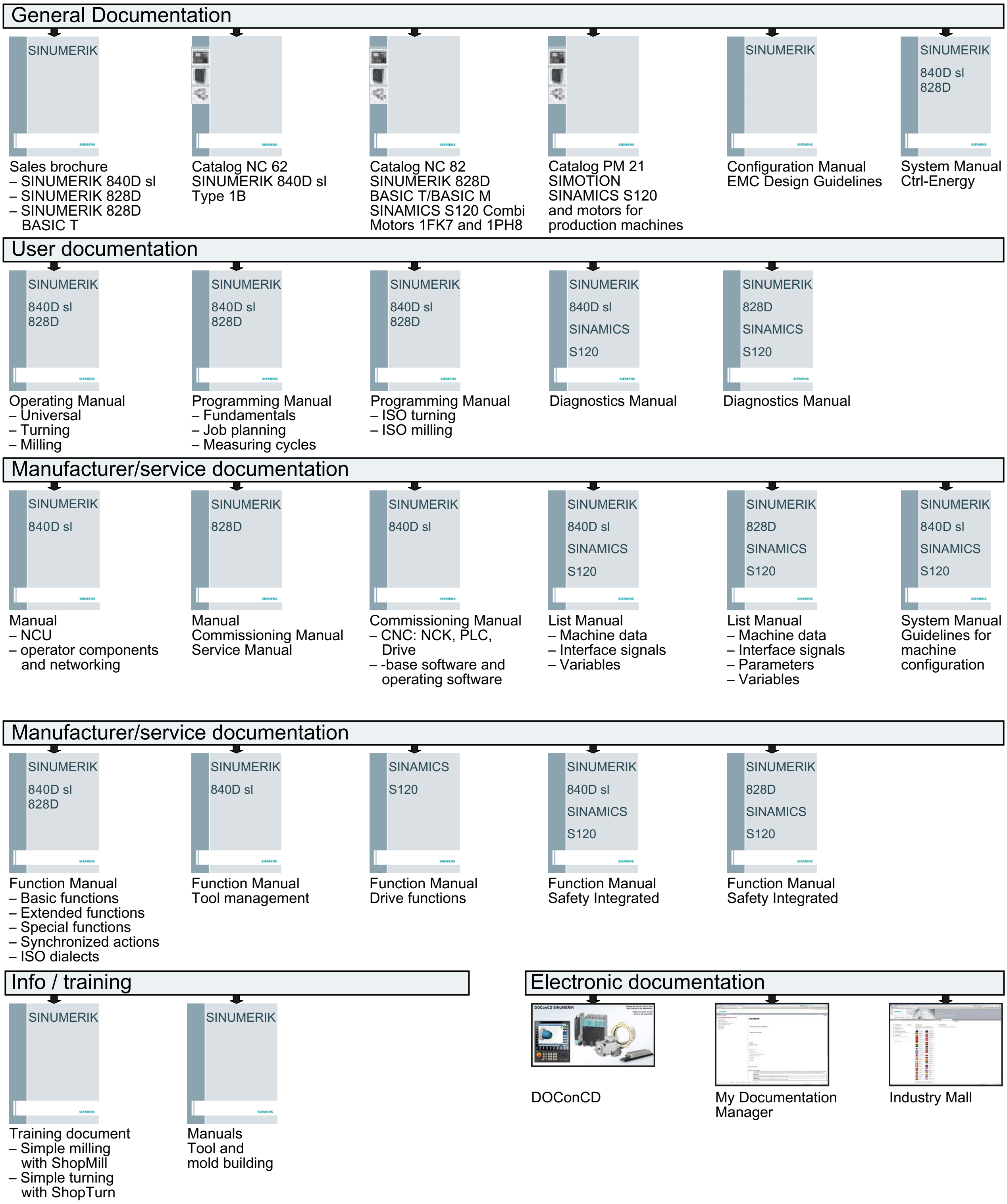
V	
VDI	Internal communication interface between NCK and PLC
VDI	Verein Deutscher Ingenieure [Association of German Engineers]
VDE	Verband Deutscher Elektrotechniker [Association of German Electrical Engineers]
VI	Voltage Input
VO	Voltage Output
FDD	Feed Drive

W	
SAR	Smooth Approach and Retraction
WCS	Workpiece Coordinate System
T	Tool
TLC	Tool Length Compensation
WOP	Workshop-Oriented Programming
WPD	Workpiece Directory: Workpiece directory
TRC	Tool Radius Compensation
T	Tool
TO	Tool Offset
TM	Tool Management
TC	Tool change

X	
XML	Extensible Markup Language

Z	
ZOA	Zero Offset Active: Identifier for zero offsets
ZSW	Status word (of drive)

A.2 Overview


General Documentation
SINUMERIK
Sales brochure
– SINUMERIK 840D sl
– SINUMERIK 828D
– SINUMERIK 828D BASIC T
Catalog NC 62
SINUMERIK 840D sl
Type 1B
Catalog NC 82
SINUMERIK 828D
BASIC T/BASIC M
SINAMICS S120 Combi
Motors 1FK7 and 1PH8
Catalog PM 21
SIMOTION
SINAMICS S120
and motors for
production machines
SINUMERIK
Configuration Manual
EMC Design Guidelines
SINUMERIK
840D sl
828D
System Manual
Ctrl-Energy
User documentation
SINUMERIK
840D sl
828D
Operating Manual
– Universal
– Turning
– Milling
SINUMERIK
840D sl
828D
Programming Manual
– Fundamentals
– Job planning
– Measuring cycles
SINUMERIK
840D sl
828D
Programming Manual
– ISO turning
– ISO milling
SINUMERIK
840D sl
SINAMICS
S120
Diagnostics Manual
SINUMERIK
828D
SINAMICS
S120
Diagnostics Manual
Manufacturer/service documentation
SINUMERIK
840D sl
Manual
– NCU
– operator components and networking
SINUMERIK
828D
Manual
Commissioning Manual
Service Manual
SINUMERIK
840D sl
Commissioning Manual
– CNC: NCK, PLC, Drive
– -base software and operating software
SINUMERIK
840D sl
SINAMICS
S120
List Manual
– Machine data
– Interface signals
– Variables
SINUMERIK
828D
SINAMICS
S120
List Manual
– Machine data
– Interface signals
– Parameters
– Variables
SINUMERIK
840D sl
SINAMICS
S120
System Manual
Guidelines for
machine
configuration
Manufacturer/service documentation
SINUMERIK
840D sl
828D
Function Manual
– Basic functions
– Extended functions
– Special functions
– Synchronized actions
– ISO dialects
SINUMERIK
840D sl
Function Manual
Tool management
SINAMICS
S120
Function Manual
Drive functions
SINUMERIK
840D sl
SINAMICS
S120
Function Manual
Safety Integrated
SINUMERIK
828D
SINAMICS
S120
Function Manual
Safety Integrated
Info / training
SINUMERIK
Training document
– Simple milling with ShopMill
– Simple turning with ShopTurn
SINUMERIK
Manuals
Tool and
mold building
Electronic documentation
DOConCD
My Documentation Manager
Industry Mall

Glossary

Absolute dimensions

A destination for an axis motion is defined by a dimension that refers to the origin of the currently active coordinate system. See → Incremental dimension

Acceleration with jerk limitation

In order to optimize the acceleration response of the machine whilst simultaneously protecting the mechanical components, it is possible to switch over in the machining program between abrupt acceleration and continuous (jerk-free) acceleration.

Address

An address is the identifier for a certain operand or operand range, e.g. input, output, etc.

Alarms

All → messages and alarms are displayed on the operator panel in plain text with date and time and the corresponding symbol for the 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 the PLC:

   Alarms and messages for the machine can be displayed in plain text from the PLC program. No additional function block packages are required for this purpose.

Archiving

Reading out of files and/or directories on an external memory device.

Asynchronous subprogram

Part program that can be started asynchronously to (independently of) the current program status using an interrupt signal (e.g. "Rapid NC input" signal).

Automatic

Operating mode of the controller (block sequence operation according to DIN): Operating mode for NC systems in which a → subprogram is selected and executed continuously.

Auxiliary functions

Auxiliary functions enable → part programs to transfer → parameters to the → PLC, which then trigger reactions defined by the machine manufacturer.

Axes

In accordance with their functional scope, the CNC axes are subdivided into:

- Axes: Interpolating path axes
- Auxiliary axes: Non-interpolating feed and positioning axes with an axis-specific feedrate. Auxiliary axes are not involved in actual machining, e.g. tool feeder, tool magazine.

Axis address

See → Axis name

Axis name

To ensure clear identification, all channel and → machine axes of the control system must be designated with unique names in the channel and control system. The → geometry axes are called X, Y, Z. The rotary axes rotating around the geometry axes → are called A, B, C.

Backlash compensation

Compensation for a mechanical machine backlash, e.g. backlash on reversal for ball screws. Backlash compensation can be entered separately for each axis.

Backup battery

The backup battery ensures that the → user program in the → CPU is stored so that it is safe from power failure and so that specified data areas and bit memory, timers and counters are stored retentively.

Basic axis

Axis whose setpoint or actual value position forms the basis of the calculation of a compensation value.

Basic Coordinate System

Cartesian coordinate system which is mapped by transformation onto the machine coordinate system.

The programmer uses axis names of the basic coordinate system in the → part program. The basic coordinate system exists parallel to the → machine coordinate system if no → transformation is active. The difference lies in the → axis names.

Baud rate

Rate of data transfer (bits/s).

Blank

Workpiece as it is before it is machined.

Block

"Block" is the term given to any files required for creating and processing programs.

Block search

For debugging purposes or following a program abort, the "Block search" function can be used to select any location in the part program at which the program is to be started or resumed.

Booting

Loading the system program after power ON.

C axis

Axis around which the tool spindle describes a controlled rotational and positioning motion.

C spline

The C spline is the most well-known and widely used spline. The transitions at the interpolation points are continuous, both tangentially and in terms of curvature. 3rd order polynomials are used.

Channel

A channel is characterized by the fact that it can process a → part program independently of other channels. A channel exclusively controls the axes and spindles assigned to it. Part program runs of different channels can be coordinated through → synchronization.

Circular interpolation

The → tool moves on a circle between specified points on the contour at a given feedrate, and the workpiece is thereby machined.

CNC

See → NC

COM

Component of the NC for the implementation and coordination of communication.

Compensation axis

Axis with a setpoint or actual value modified by the compensation value

Compensation table

Table containing interpolation points. It provides the compensation values of the compensation axis for selected positions on the basic axis.

Compensation value

Difference between the axis position measured by the encoder and the desired, programmed axis position.

Continuous-path mode

The objective of continuous-path mode is to avoid substantial deceleration of the → path axes at the part program block boundaries and to change to the next block at as close to the same path velocity as possible.

Contour

Contour of the → workpiece

Contour monitoring

The following error is monitored within a definable tolerance band as a measure of contour accuracy. An unacceptably high following error can cause the drive to become overloaded, for example. In such cases, an alarm is output and the axes are stopped.

Coordinate system

See → Machine coordinate system, → Workpiece coordinate system

CPU

Central processing unit, see → PLC

Curvature

The curvature k of a contour is the inverse of radius r of the nestling circle in a contour point (k = 1/r).

Cycles

Protected subprograms for execution of repetitive machining operations on the → workpiece.

Data block

1. Data unit of the → PLC that → HIGHSTEP programs can access.
2. Data unit of the → NC: Data modules contain data definitions for global user data. This data can be initialized directly when it is defined.

Data word

Two-byte data unit within a → data block.

Diagnostics

1. Operating area of the controller.
2. The controller has a self-diagnostics program as well as test functions for servicing purposes: status, alarm, and service displays

Dimensions specification, metric and inches

Position and pitch values can be programmed in inches in the machining program. Irrespective of the programmable dimensions (`G70`/`G71`), the 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 controller responds as if it had reached its precise destination point. A block advance of the → part program occurs.

External zero offset

Zero offset specified by the → PLC.

Fast retraction from the contour

When an interrupt occurs, a motion can be initiated via the CNC machining program, enabling the tool to be quickly retracted from the workpiece contour that is currently being machined. The retraction angle and the distance retracted can also be parameterized. An interrupt routine can also be executed following the fast retraction.

Feed override

The programmed velocity is overriden by the current velocity setting made via the → machine control panel or from the → PLC (0 to 200%). The feedrate can also be corrected by a programmable percentage factor (1 to 200%) in the machining program.

Finished-part contour

Contour of the finished workpiece. See → Raw part.

Fixed machine point

Point that is uniquely defined by the machine tool, e.g. machine reference point.

Fixed-point approach

Machine tools can approach fixed points such as a tool change point, loading point, pallet change point, etc. in a defined way. The coordinates of these points are stored in the controller. The controller 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

The geometry axes form the 2 or 3-dimensional → workpiece coordinate system in which, in → part programs, the geometry of the workpiece is programmed.

Ground

Ground is taken as the total of all linked inactive parts of a device which will not become live with a dangerous contact voltage even in the event of a malfunction.

Helical interpolation

The helical interpolation function is ideal for machining internal and external threads using form milling cutters and for milling lubrication grooves.

The helix comprises two motions:

- Circular motion in one plane
- A linear motion perpendicular to this plane

High-level CNC language

The high-level language 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). High-speed, program-driven switching functions can be initiated via the digital CNC outputs

HIGHSTEP

Summary of programming options for → PLCs of the AS300/AS400 system.

HW Config

SIMATIC S7 tool for the configuration and parameterization of hardware components within an S7 project

Identifier

In accordance with DIN 66025, words are supplemented using identifiers (names) for variables (arithmetic variables, system variables, user variables), subprograms, key words, and words with multiple address letters. These supplements have the same meaning as the words with respect to block format. Identifiers must be unique. It is not permissible to use the same identifier for different objects.

Inch measuring system

Measuring system which defines distances in inches and fractions of inches.

Inclined surface machining

Drilling and milling operations on workpiece surfaces that do not lie in the coordinate planes of the machine can be performed easily using the function "inclined-surface machining".

Increment

Travel path length specification based on number of increments. The number of increments can be stored as → setting data or be selected by means of a suitably labeled key (i.e. 10, 100, 1000, 10000).

Incremental dimension

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 controller 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 (SSFK, MSFK).

Interrupt routine

Interrupt routines are special → subprograms that can be started by events (external signals) in the machining process. A part program block which is currently being worked through is interrupted and the position of the axes at the point of interruption is automatically saved.

Inverse-time feedrate

The time required for the path of a block to be traversed can also be programmed for the axis motion instead of the feed velocity (G93).

JOG

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 controller. The key switch has three different colored keys that can be removed in the specified positions.

Keywords

Words with specified notation that have a defined meaning in the programming language for → part programs.

KÜ

Transformation ratio

KV

Servo gain factor, a control variable in a control loop.

Leading axis

The leading axis is the → gantry axis that exists from the point of view of the operator and programmer and, thus, can be influenced like a standard NC axis.

Leadscrew error compensation

Compensation for the mechanical inaccuracies of a leadscrew participating in the feed. The controller uses stored deviation values for the compensation.

Limit speed

Maximum/minimum (spindle) speed: The maximum speed of a spindle can be limited by specifying machine data, the → PLC or → setting data.

Linear axis

In contrast to a rotary axis, a linear axis describes a straight line.

Linear interpolation

The tool travels along a straight line to the destination point while machining the workpiece.

Load memory

The load memory is the same as the → working memory for the CPU 314 of the → PLC.

Look Ahead

The **Look Ahead** function is used to achieve an optimal machining speed by looking ahead over an assignable number of traversing blocks.

Machine axes

Physically existent axes on the machine tool.

Machine control panel

An operator panel on a machine tool with operating elements such as keys, rotary switches, etc., and simple indicators such as LEDs. It is used to directly influence the machine tool via the PLC.

Machine coordinate system

A coordinate system, which is related to the axes of the machine tool.

Machine zero

Fixed point of the machine tool to which all (derived) measuring systems can be traced back.

Machining channel

A channel structure can be used to shorten idle times by means of parallel motion sequences, e.g. moving a loading gantry simultaneously with machining. Here, a CNC channel must be regarded as a separate CNC control system with decoding, block preparation and interpolation.

Macro techniques

Grouping of a set of statements under a single identifier. The identifier represents the set of consolidated statements in the program.

Main block

A block prefixed by ":" introductory block, containing all the parameters required to start execution of a -> part program.

Main program

The term "main program" has its origins during the time when part programs were split strictly into main and → subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program in the channel can be selected and started. It then runs through in → program level 0 (main program level). Further part programs or → cycles as subprograms can be called up in the main program.

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 subprogram can be input and executed immediately afterwards through actuation of the NC start key.

Messages

All messages programmed in the part program and → alarms detected by the system are displayed on the operator panel in plain text with date and time and the corresponding symbol for the 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

An operating concept on a SINUMERIK controller. The following modes are defined: → Jog, → MDA, → Automatic.

Mode group

Axes and spindles that are technologically related can be combined into one mode group. Axes/spindles of a mode group can be controlled by one or more → channels. The same → mode type is always assigned to the channels of the mode group.

NC

Numerical Control: Numerical control (NC) includes all components of machine tool control: → NCK, → PLC, HMI, → COM.

Note

A more correct term for SINUMERIK controllers 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.

Network

A network is the connection of multiple S7-300 and other end devices, e.g. a programming device via a → connecting cable. A data exchange takes place over the network between the connected devices.

NRK

Numeric robotic kernel (operating system of → NCK)

NURBS

The motion control and path interpolation that occurs within the controller is performed based on NURBS (**N**on **U**niform **R**ational **B**-**S**plines). This provides a uniform procedure for all internal interpolations.

OEM

The scope for implementing individual solutions (OEM applications) has been provided for machine manufacturers, who wish to create their own user interface or integrate technology-specific functions in the controller.

Offset memory

Data range in the control, in which the tool offset data is stored.

Oriented spindle stop

Stops the workpiece spindle in a specified angular position, e.g. in order to perform additional machining at a particular location.

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:

- → Working 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

Series of statements to the NC that act in concert to produce a particular → workpiece. Likewise, this term applies to execution of a particular machining operation on a given → raw part.

Part program block

Part of a → part program that is demarcated by a line feed. There are two types: → main blocks and → subblocks.

Part program management

Part program management can be organized by → workpieces. The size of the user memory determines the number of programs and the amount of data that can be managed. Each file (programs and data) can be given a name consisting of a maximum of 24 alphanumeric characters.

Path axis

Path axes include all machining axes of the → channel that are controlled by the → interpolator in such a way that they start, accelerate, stop, and reach their end point simultaneously.

Path feedrate

Path feed affects → path axes. It represents the geometric sum of the feedrates of the → geometry axes involved.

Path velocity

The maximum programmable path velocity depends on the input resolution. For example, with a resolution of 0.1 mm the maximum programmable path velocity is 1000 m/min.

PCIN data transfer program

PCIN is 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**ontroller: → Programmable logic controller. Component of → NC: Programmable control for processing the control logic of the machine tool.

PLC program memory

SINUMERIK 840D sl: The PLC user program, the user data and the basic PLC program are stored together in the PLC user memory.

PLC programming

The PLC is programmed using the **STEP 7** software. The STEP 7 programming software is based on the **WINDOWS** standard operating system and contains the STEP 5 programming functions with innovative enhancements.

Polar coordinates

A coordinate system which defines the position of a point on a plane in terms of its distance from the origin and the angle formed by the radius vector with a defined axis.

Polynomial interpolation

Polynomial interpolation enables a wide variety of curve characteristics to be generated, such as **straight line, parabolic, exponential functions** (SINUMERIK 840D sl).

Positioning axis

Axis that performs an auxiliary motion on a machine tool (e.g. tool magazine, pallet transport). Positioning axes are axes that do not interpolate with → path axes.

Position-time cams

The term "position-time cam" refers to a pair of software cams that can supply a pulse of a certain duration at a defined axis position.

Pre-coincidence

Block change occurs already when the path distance approaches an amount equal to a specifiable delta of the end position.

Program block

Program blocks contain the main program and subprograms of → part programs.

Program level

A part program started in the channel runs as a → main program on program level 0 (main program level). Any part program called up in the main program runs as a → subprogram on a program level 1 ... n of its own.

Programmable frames

Programmable → frames enable dynamic definition of new coordinate system output points while the part program is being executed. A distinction is made between absolute definition using a new frame and additive definition with reference to an existing starting point.

Programmable logic controller

Programmable logic controllers (PLCs) are electronic controllers, the function of which is stored as a program in the control unit. This means that the layout and wiring of the device do not depend on the function of the controller. The programmable logic control has the same structure as a computer; it consists of a CPU (central module) with memory, input/output modules and an internal bus system. The peripherals and the programming language are matched to the requirements of the control technology.

Programmable working area limitation

Limitation of the motion space of the tool to a space defined by programmed limitations.

Programming key

Characters and character strings that have a defined meaning in the programming language for → part programs.

Protection zone

Three-dimensional zone within the → working area into which the tool tip must not pass.

Quadrant error compensation

Contour errors at quadrant transitions, which arise as a result of changing friction conditions on the guideways, can be virtually entirely eliminated with the quadrant error compensation. Parameterization of the quadrant error compensation is performed by means of a circuit test.

R parameters

Arithmetic parameter that can be set or queried by the programmer of the → part program for any purpose in the program.

Rapid traverse

The highest traverse 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".

RS-232-C

Serial interface for data input/output. Machining programs as well as manufacturer and user data can be loaded and saved via this interface.

Safety functions

The controller is equipped with permanently active monitoring functions that detect faults in the → CNC, the → PLC, and the machine in a timely manner so that damage to the workpiece, tool, or machine is largely prevented. In the event of a fault, the machining operation is interrupted and the drives stopped. The cause of the malfunction is logged and output as an alarm. At the same time, the PLC is notified that a CNC alarm has been triggered.

Scaling

Component of a → frame that implements axis-specific scale modifications.

Setting data

Data which communicates the properties of the machine tool to the NC as defined by the system software.

Softkey

A key, whose name appears on an area of the screen. The choice of softkeys displayed is dynamically adapted to the operating situation. The freely assignable function keys (softkeys) are assigned defined functions in the software.

Software limit switch

Software limit switches limit the traversing range of an axis and prevent an abrupt stop of the slide at the hardware limit switch. Two value pairs can be specified for each axis and activated separately by means of the → PLC.

Spline interpolation

With spline interpolation, the controller can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

Standard cycles

Standard cycles are provided for machining operations which are frequently repeated:

- For the drilling/milling technology
- For turning technology

The available cycles are listed in the "Cycle support" menu in the "Program" operating area. Once the desired machining cycle has been selected, the parameters required for assigning values are displayed in plain text.

Subblock

Block preceded by "N" containing information for a sequence, e.g. positional data.

Subprogram

The term "subprogram" has its origins during the time when part programs were split strictly into →main and subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program or any → cycle can be called up as a subprogram within another part program. It then runs through in the next → program level (x+1) (subprogram level (x+1)).

Synchronization

Statements in → part programs for coordination of sequences in different → channels at certain machining points.

Synchronized actions

1. Auxiliary function output

   During workpiece machining, technological functions (→ auxiliary functions) can be output from the CNC program to the PLC. For example, these auxiliary functions are used to control additional equipment for the machine tool, such as quills, grabbers, clamping chucks, etc.

2. Fast auxiliary function output

   For time-critical switching functions, the acknowledgement times for the → auxiliary functions can be minimized and unnecessary hold points in the machining process can be avoided.

Synchronized axes

Synchronized axes take the same time to traverse their path as the geometry axes take for their path.

Synchronized axis

A synchronized axis is the → gantry axis whose set position is continuously derived from the motion of the → leading axis and is, thus, moved synchronously with the leading axis. From the point of view of the programmer and operator, the synchronized axis "does not exist".

System memory

The system memory is a memory in the CPU in which the following data is stored:

- Data required by the operating system
- The operands timers, counters, markers

System variable

A variable that exists without any input from the programmer of a → part program. It is defined by a data type and the variable name preceded by the character **$**. See → User-defined variable.

Tapping without compensating chuck

This function allows threads to be tapped without a compensating chuck. By using the interpolating method of the spindle as a rotary axis and the drilling axis, threads can be cut to a precise final drilling depth, e.g. for blind hole threads (requirement: spindles in axis operation).

Text editor

See → Editor

TOA area

The TOA area includes all tool and magazine data. By default, this area coincides with the → channel area with regard to the access of the data. However, machine data can be used to specify that multiple channels share one → TOA unit so that common tool management data is then available to these channels.

TOA unit

Each → TOA area can have more than one TOA unit. The number of possible TOA units is limited by the maximum number of active → channels. A TOA unit includes exactly one tool data block and one magazine data block. In addition, a TOA unit can also contain a toolholder data block (optional).

Tool

Active part on the machine tool that implements machining (e.g. turning tool, milling tool, drill, LASER beam, etc.).

Tool nose radius compensation

Contour programming assumes that the tool is pointed. Because this is not actually the case in practice, the curvature radius of the tool used must be communicated to the controller which then takes it into account. The curvature center is maintained equidistantly around the contour, offset by the curvature radius.

Tool offset

Consideration of the tool dimensions in calculating the path.

Tool radius compensation

To directly program a desired → workpiece contour, the control must traverse an equistant path to the programmed contour taking into account the radius of the tool that is being used (`G41`/`G42`).

Transformation

Additive or absolute zero offset of an axis.

Travel range

The maximum permissible travel range for linear axes is ± 9 decades. The absolute value depends on the selected input and position control resolution and the unit of measurement (inch or metric).

User interface

The user interface (UI) is the display medium for a CNC in the form of a screen. It features horizontal and vertical softkeys.

User memory

All programs and data, such as part programs, subprograms, comments, tool offsets, and 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

The working memory is a RAM in the → CPU that the processor accesses when processing the application program.

Workpiece

Part to be made/machined by the machine tool.

Workpiece contour

Set contour of the → workpiece to be created or machined.

Workpiece coordinate system

The workpiece coordinate system has its starting point in the → workpiece zero-point. In machining operations programmed in the workpiece coordinate system, the dimensions and directions refer to this system.

Workpiece zero

The workpiece zero is the starting point for the → workpiece coordinate system. It is defined in terms of distances to the → machine zero.

Zero offset

Specifies a new reference point for a coordinate system through reference to an existing zero point and a → frame.

1. Settable

   A configurable number of settable zero offsets are available for each CNC axis. The offsets - which are selected by means of G functions - take effect alternatively.

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

$

1

2

3

4

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z