Description of Functions 10/2004 Edition

sinumerik

SINUMERIK 840D/840Di/810D
Extended Functions

SIEMENS

SIEMENS

# SINUMERIK 840D/840Di/ SINUMERIK 810D

# Extended Functions

## Description of Functions

**Valid for**

| *Control* | *Software Version* |
|---|---|
| SINUMERIK 840D powerline | 7 |
| SINUMERIK 840DE powerline | 7 |
| SINUMERIK 840Di | 3 |
| SINUMERIK 840DiE (export version) | 3 |
| SINUMERIK 810D powerline | 7 |
| SINUMERIK 810DE powerline (exp.) | 7 |

**10.2004 Edition**

# SINUMERIK® Documentation

## Printing history

Brief details of this edition and previous editions are listed below.

The status of each edition is shown by the code in the “Remarks” columns.

*Status code in the “Remarks” column:*

**A** . . . . . New documentation.
**B** . . . . . Unrevised reprint with new Order No.
**C** . . . . . Revised edition with new status.
If factual changes have been made on the page since the last edition, this is indicated by a new edition coding in the header on that page.

| Edition | Order No. | Remarks |
|---|---|---|
| 06.94 | 6FC5297-0AC30-0BP0 | A |
| 08.94 | 6FC5297-0AC30-0BP1 | C |
| 02.95 | 6FC5297-2AC30-0BP0 | C |
| 04.95 | 6FC5297-2AC30-0BP1 | C |
| 03.96 | 6FC5297-3AC30-0BP0 | C |
| 08.97 | 6FC5297-4AC30-0BP0 | C |
| 12.97 | 6FC5297-4AC30-0BP1 | C |
| 12.98 | 6FC5297-5AC30-0BP0 | C |
| 08.99 | 6FC5297-5AC30-0BP1 | C |
| 04.00 | 6FC5297-5AC30-0BP2 | C |
| 10.00 | 6FC5297-6AC30-0BP0 | C |
| 09.01 | 6FC5297-6AC30-0BP1 | C |
| 11.02 | 6FC5297-6AC30-0BP2 | C |
| 03.04 | 6FC5297-7AC30-0BP0 | C |
| 10.04 | 6FC5297-7AC30-0BP1 | C |



Further information is available in the Internet at:
http://www.siemens.com/motioncontrol

This publication was produced with Interleaf V 7





Order No. 6FC5297-7AC30-0BP1
Printed in Germany

Siemens Aktiengesellschaft

# Preface

**Notes for the reader**

The SINUMERIK documentation is subdivided into 4 parts:

- General Documentation
- User Documentation
- Manufacturer/Service documentation
- OEM documentation

This document is designed for machine tool manufacturers. It contains a detailed description of the scope of functions offered by SINUMERIK controls.

The function descriptions are only valid for the specific software version or up to the software version specified. You should request valid function descriptions for new software versions. Old function descriptions are only partly applicable for new software versions.

More detailed information about other SINUMERIK 840D/840Di/810D documents and publications for all SINUMERIK controllers (e.g. universal interface, measuring cycles, etc.) can be obtained from your local Siemens representative.

---

**Note**

It may be possible to run functions that are not described in this document in your controller. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

---

**Hotline**

If you have any questions about the control, please contact the following hotline:

A&D Technical Support Tel.: +49 (180) 5050 222
Fax: +49 (180) 5050-223
Internet: http://www.siemens.com/automation/support-request

If you have any queries about the documentation (suggestions, corrections), please fax them to the following number:
Fax: +49 (9131) 98-2176
E-mail: motioncontrol.docu@erlf.siemens.de

Fax form: See the reply form at the end of the brochure

**SINUMERIK Internet address**

http://www.siemens.com/motioncontrol

**SINUMERIK 840D powerline**

Since 09.2001 the

- SINUMERIK 840D powerline and
- SINUMERIK 840DE powerline

have been available with improved performance. A list of available **powerline** modules can be found in Section 1.1 of the Hardware Reference Manual /PHD/.

**SINUMERIK 810D powerline**

Since 12.2001 the

- SINUMERIK 810D powerline and
- SINUMERIK 810DE powerline

have been available with improved performance. A list of available **powerline** modules can be found in Section 1.1 of the Hardware Reference Manual /PHC/.

**Objective**

The function descriptions provide the information required for configuration and installation.

**Target groups**

The information contained in the function descriptions is designed for:

- Design engineers
- PLC programmers creating the PLC user program with the signals listed
- Start-up engineers once the system has been configured and set up
- Maintenance personnel inspecting and interpreting status signals and alarms

**Notes on how to use this manual**

This Description of Functions is structured as follows:

- General table of contents (overview) of the manual
- Descriptions of functions in alphanumeric order of the Description of Function codes
- Appendix with keyword index

---

**Note**

The Description of Functions **Basic Machine** (Part 1) contains both a general index as well as a reference list, a glossary of terms used and a list of abbreviations and acronyms.

Pages indicated provide the following information:
Part of the Description of Functions / Book / Chapter / Section / Subsection – Page

---

If you need information about a particular function, you will find the function and the code under which it is sorted in the inside title page of the manual.

If you need information about a particular term, please look for the term in the section headed Index in the Appendix. The Description of Functions code, the chapter number and the number of the page on which you can find the information you need are listed in this section.

Chapters 4 and 5 of each Description of Functions contain definitions for "Active, data format, input limits", etc. for the various signals and data.
An explanation of these definitions appears below under "Technical information".

---

!

**Important**

This documentation applies to:

- SINUMERIK 840D powerline control, software version 7
- SINUMERIK 810D powerline control, software version 7
- SINUMERIK 840Di control, software version 2

---

**Equivalent software versions**

The software versions specified in this documentation refer to the SINUMERIK 840D powerline and 810D powerline control systems with the parallel applicable software version. Whether individual functions have been approved for the control systems is not specified explicitly in each case, for further details, see /BU/, Catalog NC 60. The following applies:

Table 1-1 Equivalent software version

| SINUMERIK 840D powerline | | SINUMERIK 810D powerline | SINUMERIK 840Di |
|---|---|---|---|
| 7.1 | equivalent to | 7.1 | 3.1 |

## Explanation of symbols

---

**Important**

This symbol is always displayed in this document to draw your attention to an important item of information.

---

**Ordering Data Option**

In this documentation you will find the symbol shown on the left with a reference to an ordering data option. The described function is only executable on the control if the control has the designated option.

---

**Machine Manufacturer**

This pictorial symbol appears in this document to indicate that the machine manufacturer can control or modify the function described. See machine manufacturer's specifications.

---

**Danger**

Indicates an imminently hazardous situation which, if not avoided, **will** result in death or serious injury or in substantial property damage.

---

**Warning**

Indicates a potentially hazardous situation which, if not avoided, **could** result in death or serious injury or in substantial property damage.

---

**Caution**

Used with the safety alert symbol indicates a potentially hazardous situation which, if not avoided, **may** result in minor or moderate injury or in property damage.

---

**Caution**

Used without safety alert symbol indicates a potentially hazardous situation which, if not avoided, **may** result in property damage.

**Notice**

Used without the safety alert symbol indicates a potential situation which, if not avoided, **may** result in an undesirable result or state.

## Technical information

### Notations

The following notation and abbreviations are used in this documentation:

- PLC interface signals –> IS "signal name" (signal data)
  E.g.: – IS "MMC-CPU1 ready" (DB10, DBX108.2) i.e. the signal is stored in data block 10, data byte 108, bit 2.
  – IS "Feedrate/Spindle speed override" (DB31-48, DBB0) i.e. the signals for each axis/spindle are stored in data blocks 31 to 48, data block byte 0.
- Machine data –> MD: MD_NAME (German name)
- Setting data –> SD: SD_NAME (German name)
- The symbol "≙" means "corresponds to"

### Explanation of the abbreviations in Chapters 4 and 5

Chapters 4 and 5 of each Description of Functions describe the data and signals which are significant for the respective function. Certain terms and abbreviations, which are used in these tabular descriptions, are explained here.

Values in the table

The machine data indicated in the Descriptions of Functions are always values for an NCU572.

The values for a different NCU (e.g. NCU570, NCU571, NCU573) are contained in the List Manual.

**References:** /LIS/, "Lists"

Default value

The machine data/setting data is preset to this value during start-up. In cases where different default values are used for different channels, this is indicated by a " / ".

Value range (minimum and maximum)

Specifies the input limits. If no value range is specified, the data type determines the input limits and the field is marked "***".

When changes take effect

When machine data, setting data, etc. are altered, they are not immediately active. The conditions for activation are therefore always specified. The possibilities are shown in the following list in order of their priority:

- POWER ON (po) "RESET" key on front panel of the NCU module or switch power supply on/off
- NEW_CONF (cf) – Reconfiguration of the PLC interface
  – "RESET" on control unit, or
- RESET (re) "RESET" key on control unit or
- Immediately (im) after entry of the value

Protection level

Protection levels 0 to 7 have been used. The lock for protection levels 0 to 3 (4 to 7) can be canceled by entering the correct password (setting correct keyswitch position). The operator only has access to information protected by one particular level and the levels below it. The machine data is assigned different protection levels as a standard measure.

The table contains only the protection level for write operations. A fixed relationship between read and write levels exists however:

| Protection level for writing | Protection level for reading |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 4 |

**References:** /BA/, "Operator's Guide"
/FB/, A2, "Various Interface Signals"

Unit

The unit refers to the default setting of machine data SCALING_FACTOR_USER_DEF_MASK and
SCALING_FACTOR_USER_DEF.
If the MD has no underlying unit, a "–" is shown in the box.

Data type

The following data types are used in the control system:

- DOUBLE
  Real or integer values (decimal values or whole numbers)
  Input limits from +/–4.19*$10^{-307}$ to +/–1.67*$10^{308}$
- DWORD
  Integer values
  Input limits from –2.147*$10^{9}$ to +2.147*$10^{9}$
- BOOLEAN
  Possible input values: true or false / 0 or 1
- BYTE
  Integer values from –128 to +127
- STRING
  Consisting of max. 16 ASCII characters (upper case letters, numbers and underscore)

**Data management**

The descriptions of the PLC interface in the individual Descriptions of Functions assume a theoretical maximum number of components:

- 4 mode groups (associated signals stored in DB11)
- 8 channels (associated signals stored in DB21-30)
- 31 axes (associated signals stored in DB31-61)

For details of the actual number of components which can be implemented with each software version, please refer to
**References:** /BU/, “Order Document”, Catalog NC 60

■

# Notes

# SINUMERIK 840D/840Di/810D Description of Functions Extended Functions (FB2)

# Digital and Analog NCK I/Os (A4)

■

# 1 Brief Description

**General**

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

- Several feed values in one block
- Several auxiliary functions in a block
- Rapid retraction on finished contour
- Axis-specific delete distance-to-go
- Program branches
- Rapid NC start
- Analog calipers
- Position switching signals
- Punching/nibbling functions
- Analog value control
- etc.

**Contents**

This Description of Functions describes the specifications for the digital and analog I/Os.
The note “References” lists documentation relating to any function which utilizes these I/Os.

■

# Notes

# Detailed Description 2

## 2.1 General functionality

**General**

The ability to control or influence time-critical NC functions is dependent on high-speed NCK I/O interfaces or the facility to rapidly address particular PLC I/Os (see Section 2.5).

On the SINUMERIK 840D, 840Di system, therefore,

a) digital and analog NCK inputs and outputs can be used (see Chapter 3).

b) specific PLC I/Os can be addressed directly (see Section 2.5).

The hardware inputs and outputs can be read and written via system variables in the parts program or synchronized actions.

Via the PLC interface, both the signal states of the digital I/Os and the values of the external analog I/Os can be changed by the PLC user program according to the application.

**840D hardware**

On the SINUMERIK 840D **onboard NCU** there are 4 digital NCK inputs (inputs 1 to 4) and 4 digital NCK outputs (outputs 1 to 4).

The digital onboard inputs and outputs are stored in the first address byte. With the NCK outputs, the remaining signals of this byte (NCK outputs 5 to 8) can be used via the PLC interface (digital NCK outputs without hardware).

Using the “NCU terminal block” that can be coupled to the drive bus, it is possible to connect further digital NCK inputs/outputs and analog NCK inputs/outputs (hereafter called **external NCK I/Os**). The “NCU terminal block” is used as a carrier module for up to eight DP compact plug-in modules. Up to two “NCU terminal blocks” can be connected per NCU.

The maximum degree of expansion of the external NCK I/Os is:

- 32 digital NCK inputs (digital inputs 9 to 40)
- 32 digital NCK outputs (digital outputs 9 to 40)
- 8 analog NCK inputs (analog inputs 1 to 8)
- 8 analog NCK outputs (analog outputs 1 to 8)

For further information about the hardware specification see:

**References:** /PHD/, SINUMERIK 840D, NCU Manual

**840 Di hardware digital I/Os**

Digital inputs/outputs are provided for the SINUMERIK 840Di via the MCI Board Extension module. The following connections are available:

- 2 handwheels
- 2 probes
- 4 digital inputs/outputs

---

**Note**

The MCI Board Extension module is an option for the SINUMERIK 840Di. The PIN assignment of the cable distribution interface (X121) matches the cable distributor assignment on the SINUMERIK 840D.

---

**SINUMERIK 840Di analog and digital inputs/outputs**

Analog and digital inputs/outputs can be operated on the SINUMERIK 840Di by means of SIMATIC S7 bus interface and signal boards linked via the PROFIBUS-DP.

**PLC I/Os for direct addressing by NCK SW 5.2**

Up to 16 bytes for digital input signals and analog input values plus a total of 16 bytes for digital output signals and analog output values can be addressed directly by the parts program. These bytes must be taken into account when the PLC is configured. They must be programmed consecutively. They are processed directly by the PLC operating system. As a result, the signal transfer time between the NC and PLC I/O modules is of a magnitude of 0.5 ms.

---

**Caution**

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 details, see 2.5.

**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 in a setting data.

For more information please refer to Section 2.7.

**Number**

The number of addressable digital NCK input/output bytes and analog inputs/outputs must be programmed by means of general machine data.

| Machine data ($MN_ ... ) | Number of active ... | Max. number |
|---|---|---|
| FASTIO_DIG_NUM_INPUTS | Digital NCK input **bytes** | 5 |
| FASTIO_DIG_NUM_OUTPUTS | Digital NCK output **bytes** | 5 |
| FASTIO_ANA_NUM_INPUTS | Analog NCK inputs | 8 |
| FASTIO_ANA_NUM_OUTPUTS | Analog NCK outputs | 8 |

**Note**

The 1st byte is always assigned to the 4 digital I/Os on the MCI Board Extension module on the SINUMERIK 840Di. Even if you have not connected an MCI Board Extension module to the SINUMERIK 840Di, the 1st byte is always assigned to it.
For this reason, at least 2 bytes must always be entered in machine data FASTIO_DIG_NUM... if you want to operate further I/Os via the PROFIBUS.

Corresponding alarms are generated if the parts program addresses inputs/outputs that have not been defined in the above machine data.

These NCK inputs or outputs do not have to actually exist in the hardware. If they do not, the signal states or the binary analog values are set to "zero" in a defined way inside the NCK. The values can be changed by the PLC.

**Hardware assignment of external NCK I/Os**

The following general machine data ($MN–) are provided for assigning I/O signal modules or I/O modules to external NCK I/Os:

- MD 10366: HW_ASSIGN_DIG_FASTIN[hw] Hardware assignment for external digital inputs
- MD 10368: HW_ASSIGN_DIG_FASTOUT[hw] Hardware assignment for external digital outputs
- MD 10362: HW_ASSIGN_ANA_FASTIN[hw] Hardware assignment for external analog inputs
- MD 10364: HW_ASSIGN_ANA_FASTOUT[hw] Hardware assignment for external analog outputs

[hw]: Index for addressing the external digital I/O bytes (0 to 3) or the external analog inputs/outputs (0 to 7)

**Note**

The hardware assignment is different on the SINUMERIK 840D and 840Di controls.

The defaults for the **assignment** of I/Os for the SINUMERIK 840Di via machine data MD 10362 to MD 10368 are as follows:

| Machine data ($MN_ ... ) | Meaning | Default |
|---|---|---|
| HW_ASSIGN_ANA_FASTIN[0]<br>... | Assignment for analog input<br>(16-bit access) | 050000A0 |
| HW_ASSIGN_ANA_FASTOUT[0]<br>... | Assignment for analog output<br>(16-bit access) | 050000A0 |
| HW_ASSIGN_DIG_FASTIN[0]<br>... | Assignment for digital input<br>(8-bit access) | 05000090 |
| HW_ASSIGN_DIG_FASTOUT[0]<br>... | Assignment for digital output<br>(8-bit access) | 05000090 |

**Modification to MD for PROFIBUS DP**

The machine data $MN_HW_ASSIGN_... have been modified for hardware operation on the PROFIBUS DP of the SINUMERIK 840Di.

The assignment of bytes 1 to 4 has been redefined. The machine data assignments below apply for PROFIBUS DP operation:

| Byte | New for PROFIBUS-DP | Old meaning |
|---|---|---|
| 4th byte | Segment number = 5 | Segment number |
| 3rd byte | Not used = 0 | Module number |
| 2nd byte | Logical address high | Submodule number |
| 1st byte | Logical address low | Input/output number |

**Guidelines** for machine data $MN_HW_ASSIGN_...:

- Logical address in 1st and 2nd byte is specified in hexadecimal format. Example: 050001A2 (Hex) equals logical address 418 (Dec).
- Address 0 is reserved for the PLC and cannot be used as an NC I/O.
- The value 05000000 in MD $MN_HW_ASSIGN_... is interpreted as “Slot does not physically exist”. The input is then treated like a simulation input.

**System variables**

The following table lists the system variables with which NCK I/Os can be read or written directly by the parts program.

The number of the NCK input/output is used for addressing.

The following applies to n:
$1 \leq n \leq 8$ * MD 10350: FASTIO_DIG_NUM_INPUTS
$1 \leq n \leq 8$ * MD 10360: FASTIO_DIG_NUM_OUTPUTS

$1 \leq n \leq$ MD 10300: FASTIO_ANA_NUM_INPUTS
$1 \leq n \leq$ MD 10310: FASTIO_ANA_NUM_OUTPUTS

| System variable | Meaning | Range of [n] |
|---|---|---|
| $A_IN[n] | Read digital NCK input [n] | 1 to 3, 9 to 40 |
| $A_INA[n] | Read analog NCK input [n] | 1 to 8 |
| $A_INCO[n] | Read comparator input [n] | 1 to 16 |
| PBB | | |
| $A_OUT[n] | Read/write digital NCK output [n] | 1 to 40 |
| $A_OUTA[n] | Read/write analog NCK output [n] | 1 to 8 |

**Note**

When this system variable is read by the parts program, a preprocess stop (STOPRE command) is initiated inside the control.

### Weighting factor

The weighting factors in the general machine data MD 10320: FASTIO_ANA_INPUT_WEIGHT[hw] and MD 10330: FASTIO_ANA_OUTPUT_WEIGHT[hw] allow each individual analog NCK input and output to be adapted to the AD or DA converters of the analog I/O module used.

If the correct weighting factor is set, the value set in system variable $A_OUTA[n] outputs the corresponding voltage value in millivolts at the analog output [n].

Example for 840D

The analog value range is 10V (maximum modulation);
FASTIO_ANA_OUTPUT_WEIGHT[hw] = 10000 (default on 840D)
$A_OUTA[1] = 9500 ; 9.5V is output at analog NCK output 1
$A_OUTA[3] = –4120 ; –4.12V is output at analog NCK output 3

Application for analog NCK inputs/outputs without hardware:

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

### Assignment to NC functions

Several NC functions are dependent on the functionality of the NCK I/Os. The NCK inputs and/or outputs used for these functions are assigned on a function-specific basis via machine data (e.g. MD 21220: MULTFEED_ASSIGN_FASTIN for “Multiple feedrates in one block”). A byte address must be specified in the machine data for the digital inputs/outputs; the assignment is always made byte by byte.

| Byte address | Assignment for the digital NCK inputs/outputs |
|---|---|
| 0 | none |
| 840D: 1 | 1 to 4 (onboard I/Os) and 5 to 8 (NCK-A without hardware) |
| FM-NC: 1 | 1 to 8 (NCK-A 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 (see Section 2.7) |
| 129 | Inputs 9 to 16 of comparator byte 2 (see Section 2.7) |

**Clock-synchronous processing**

The I/O modules of the external NCK I/Os on the SINUMERIK 840D can be operated in one of the following two modes:

- **Asynchronously**, i.e. the input and output values are made available in cycles set by the terminal block which are asynchronous to the internal NC processing cycles.
- **Synchronously**, i.e. the input values and the output values are provided synchronously with the settable internal NC processing clock frequency.

The processing mode is selected for individual modules by means of general machine data MD 10384: HW_CLOCKED_MODULE_MASK[tb].

[tb] = Index for terminal block (0 to 1)

In synchronous processing mode, one of the following clock rates can be selected (general MD 10380: HW_UPDATE_RATE_FASTIO[tb]):

- Synchronous inputs/outputs in position control cycles (default setting)
- Synchronous inputs/outputs in interpolation cycles

It is possible to define a lead time in microseconds for the clocked NCK I/Os in general MD 10382: HW_LEAD_TIME_FASTIO[tb]. This makes it possible to consider the conversion time of the ADC for example, so that the digitized input value is available on the cycle.

The defined cycle frequency or delay time applies to all cycle-synchronous I/O modules of the terminal block addressed with [tb].

On the SINUMERIK FM-NC, the I/O modules of the external NCK I/Os **always** operate asychronously. They are updated in position control cycles.

**Monitoring functions**

The following functional monitors are provided for external I/Os on the SINUMERIK 840D:

- During booting:
  - Check whether the I/O modules in the terminal blocks match 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 “safe” status (i.e. 0V at output) in the event of faults (e.g. NC ready = 0) in the NCU or power failures.

## 2.2 Digital inputs/outputs of the NCK

### 2.2.1 Digital inputs of the NCK

**Number**

General MD 10350: FASTIO_DIG_NUM_INPUTS (number of active digital NCK input bytes) the available digital NCK inputs can be defined (in groups of 8).

**Function**

The digital NCK inputs allow external signals to be injected which can then be used, for example, to control the workpiece machining program sequence. With the system variable **$A_IN[n]**, the signal status of the digital input [n] can be scanned directly in the parts program.

The signal state at the hardware input can be changed by the PLC user program (see Fig. 2-1).

**Disable input**

The PLC user program can disable NCK inputs individually by means of interface signal “Disable digital NCK inputs” (DB10, DBB0 or DB122 ...). In this case, they are set to “0” in a defined manner inside the control.

**Set input from PLC**

The PLC can also apply interface signal “Setting digital NCK inputs on PLC” (DB10, DBB1 or DBB123 ...) to set each digital input to a defined “1” signal state (see Fig. 2-1). As soon as this interface signal is set to “1”, the signal state at the hardware input or the input disable is inactive.

**Read actual value**

The signal status of the digital NCK inputs is signaled to the PLC (interface signal “Actual value of digital NCK inputs” (DB10, DBB60, DBB186 ...)). The actual value reflects the real state of the signal at the hardware input; the influence of the PLC is therefore ignored in the “actual value” (see Fig. 2-1).

**RESET/power ON behavior**

After power ON and reset, the signal level at the input is passed on. If necessary, the PLC user program can disable or set the inputs to “1” in a defined manner as described above.

**Applications**

The program sequence can be controlled with conditional jump statements in the part program as a function of the signal status of an external hardware signal.

For example, digital NCK inputs can be used 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**: /FB/, S5, “Synchronized Actions”

The NCK inputs are assigned to the NC functions separately for each function and byte in the machine data. Multiple assignments of inputs are not monitored.

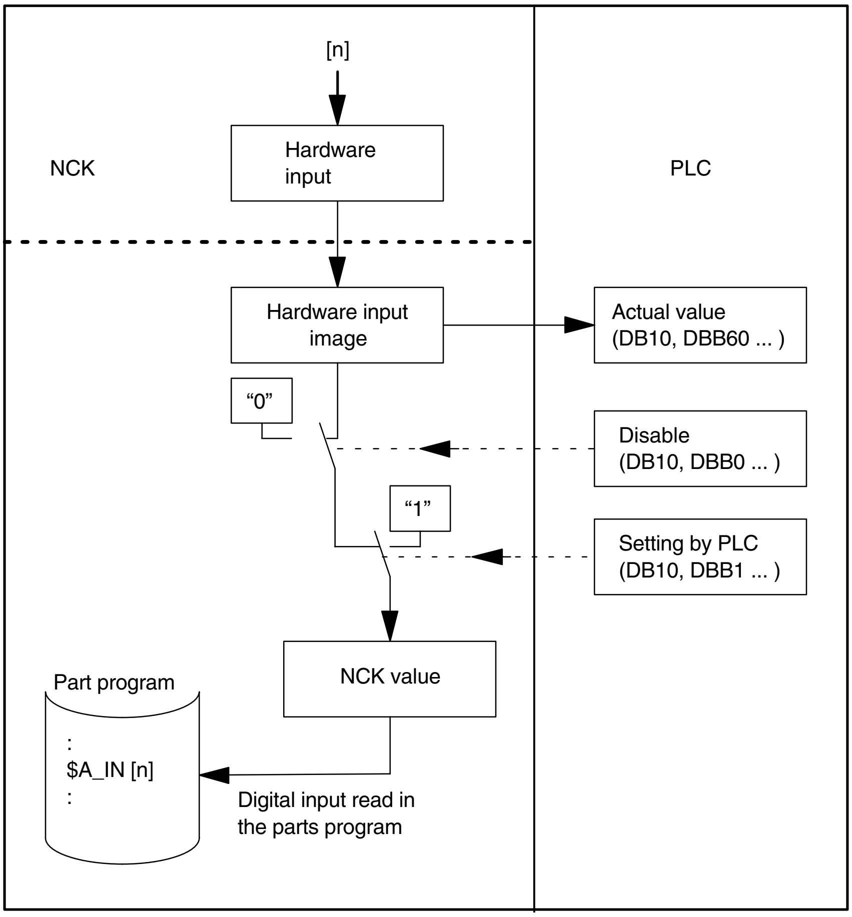



Fig. 2-1 Signal flow for digital NCK inputs

## 2.2.2 Digital outputs of the NCK

**Number**

General MD 10310: FASTIO_DIG_NUM_OUTPUTS (number of active digital NCK output bytes) the available digital NCK outputs can be defined (in groups of 8).

**Function**

The digital NCK outputs provide the option of outputting important switching commands at high speed as a function of the program processing status. With the system variable **$A_OUT[n]**, the signal status of the digital output [n] can be set or read again directly in the parts program.

There are also several ways of changing this set signal state via the PLC (see Fig. 2-2).

**Disable output**

The PLC user program is capable of disabling the digital NCK outputs individually with interface signal “Disable digital NCK outputs” (DB10, DBB4, DBB130...). In this case, the “0” signal is output at the hardware output (see Fig. 2-2).

**Overwrite screenform**

Every output that can be set by the NC parts program can be **overwritten** from the PLC using the overwrite screenform. Previous NCK values are then lost (see Fig. 2-2).

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

1. The output in question must be preset with the required signal state at the PLC interface “PLC setting for digital NCK outputs” (DB10, DBB6, DBB132...).
2. The setting value becomes the new NCK value for the relevant output (DB10, DBB5, DBB131 ...) when the overwrite screenform is activated (signal transition 0 –> 1). This value remains operative until a new NCK value is programmed (by the PLC or from the NC part program).

**Setting screen form**

Furthermore, a PLC setting for each output can determine whether the instantaneous (e.g. as specified by NC parts program) or the PLC value specified via the setting screen form (DB10, DBB7, DBB133 ...) should be sent to the hardware output (see Fig. 2-2).

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

1. The output in question must be preset with the required signal state at the PLC interface “PLC setting for digital NCK outputs” (DB10, DBB6).
2. The setting screen form must be set to “1” for the output in question.

Unlike the overwrite screenform, the current NCK value is not lost when a value is set in the setting screen form. As soon as the PLC sets “0” in the setting screen form, the NCK value is again active.

**Note**

The same setting value (DB10, DBB6) is used at the PLC interface for the overwrite and setting screenforms. Therefore, an identical output signal state is the result if the signal state is changed **simultaneously** in the overwrite **and** setting screenform.

**Read setpoint**

The instantaneous NCK value at the digital outputs can be read by the PLC user program (interface signal "setpoint of digital NCK outputs" (DB10, DBB64, DBB186 ...)). Please note that this setpoint ignores disabling and the setting screen form of the PLC. The setpoint can therefore be different from the actual signal state at the hardware output (see Fig. 2-2).

**RESET/ end of program**

On end of program or RESET, every digital output can be defined as necessary by the PLC user program in the overwrite screenform, setting screen form or disable signal.

**Power ON**

After power ON, the digital outputs are set to "0" in a defined manner. This can be overwritten in the PLC user program according to the application using the screen forms described above.

**Digital NCK outputs without hardware**

No alarm is output if the digital NCK outputs written from the part program have been defined in general MD 10360: FASTIO_ANA_NUM_INPUTS, but do not exist as hardware outputs. The NCK value can be read by the PLC (IS "Setpoint ...")

**Applications**

This function allows digital hardware outputs to be set instantaneously by bypassing the PLC cycles. Time-critical switching functions can thus be triggered in connection with the machining process and under program control (e.g. on block change).

For example digital NCK outputs are required for the following NC functions:

- Position signals
  **References:** /FB/, N3, "Software Cams, Position Signals"
- Output of the comparator signals (see Section 2.7)

The NCK outputs are assigned to the NC functions separately for each function in machine data. Multiple assignments of outputs are checked during power ON and indicated by an alarm.

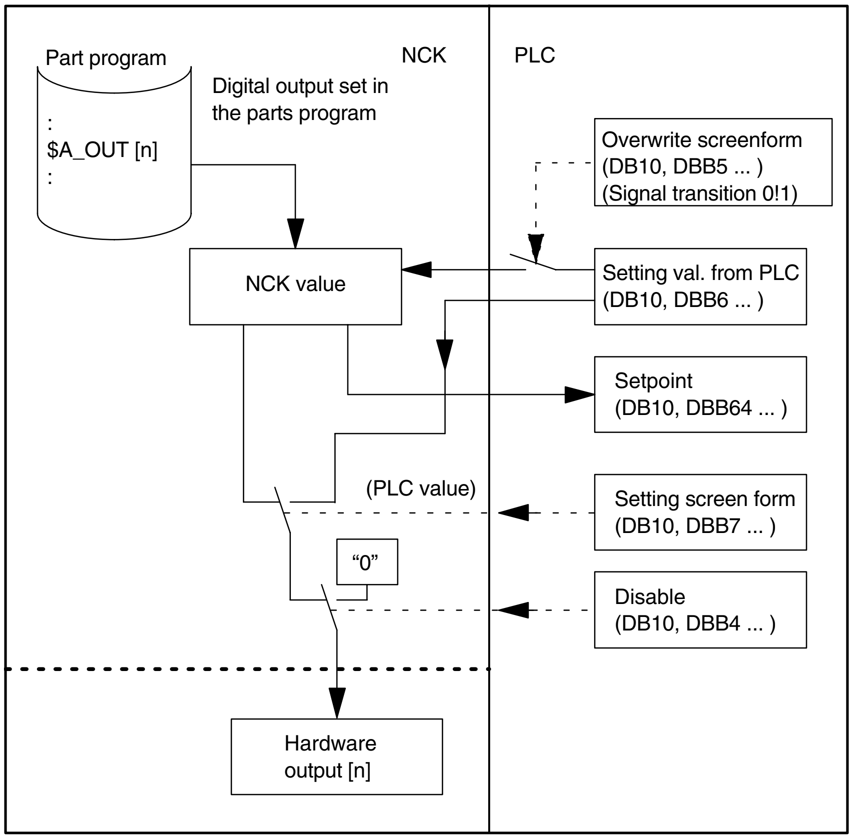


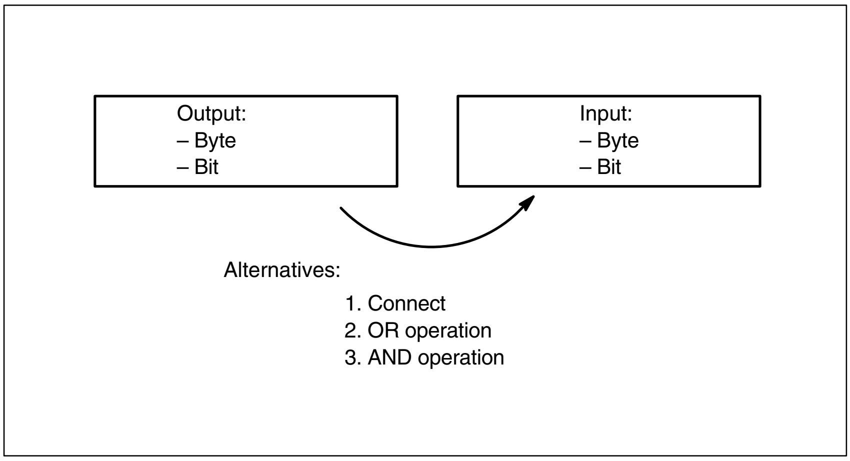

Fig. 2-2 Signal flow for digital NCK outputs

## 2.3 Connecting and logic operations of fast NCK inputs/outputs

**Function**

In SW 4 and higher, the fast inputs of the NCK I/Os can be set in the software according to the signal states of the fast outputs.

Overview:




**Connect**

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

**OR operation**

The fast input of the NCK I/O takes the signal state which is given by the OR operation of the output signal with the assigned input signal.

**AND operation**

The fast input of the NCK I/O takes the signal state which is given by ANDing 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 alarm. Checking of the active bytes of the NCK I/Os is performed via the entries in the machine data:
  MD 10350: FASTIO_DIG_NUM_INPUTS and
  MD 10360: FASTIO_DIG_NUM_OUTPUTS.

**Defining assignments**

The assignments are specified via machine data:

MD 10361 : FASTIO_DIG_SHORT_CIRCUIT[n].

n can assume values between 0 and 9, in other words, up to **10** assignments can be defined.

2 hexadecimal characters in each case are provided for the specification of the byte and bit of an output. The **type of logic operation** is specified by entering

0 for connect
A for AND operation
B...for OR operation

in bits 12 – 15 of the input.

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:
MD 10361: FASTIO_DIG_SHORT_CIRCUIT = ’04010302H’
output 4, byte 1, connect to
input 3, byte 2

AND operation:
MD 10361: FASTIO_DIG_SHORT_CIRCUIT = ’0705A201H’
output 7, byte 5 AND with
input 2, byte 1

OR operation:
MD 10361: FASTIO_DIG_SHORT_CIRCUIT = ’0103B502H’
output 1, byte 3, OR with
input 5, byte 2

## 2.4 Analog inputs/outputs of the NCK

### 2.4.1 Analog inputs of the NCK

**Number**

General MD 10300: FASTIO_ANA_NUM_INPUTS (number of active analog NCK inputs) the available analog NCK inputs can be defined.

**Function**

The system variable **$A_INA[n]** allows the value at the analog NCK input [n] to be directly accessed in the parts program.

The analog value at the hardware input can also be influenced by the PLC user program (see Fig. 2-3).

**Disable input**

The PLC user program is capable of disabling the analog NCK inputs individually with interface signal "Disable analog NCK inputs" (DB10, DBB146). In this case, they are set to "0" in a defined manner inside the control.

**Set input from PLC**

The PLC can also specify a value for each analog NCK input by applying the interface signal "Setting screen form of analog NCK inputs" (DB10, DBB147) (see Fig. 2-3). As soon as this interface signal is set to "1", the value set by the PLC (DB10, DBB148 to 163) becomes active for the analog input. The analog value at the hardware input or the input disable is then inactive.

**Read actual value**

The interface signal "Actual value of analog input of NCK" (DB10, DBB194 to 209) transfers the analog values that are actually present at the hardware inputs to the PLC. The possible influence of the PLC is therefore ignored in the actual value (see Fig. 2-3).

**RESET/power ON behavior**

After power ON and RESET, the analog value at the input is passed on. If necessary, the PLC user program can manipulate the NCK inputs as described above in the PLC user program.

**Weighting factor**

Using the weighting factor in the general MD 10320: FASTIO_ANA_INPUT_WEIGHT[hw] it is possible to adapt each analog NCK input to the various ADCs for reading in the parts program (see Fig. 2-3).

In this machine data it is necessary to enter the value x that is to be read in the parts 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 the system variable $A_INA[n] as a numerical value with the unit millivolts.

**Binary analog value display**

See Section 2.6

**Analog NCK input without hardware**

When the parts program accesses analog NCK inputs that have been defined in MD 10300: FASTIO_ANA_NUM_INPUTS but that do not exist as hardware inputs, the following values are read:

- The setpoint set by the PLC if the IS “PLC setting for analog NCK inputs” is set to “1” (see Fig. 2-3)
- Otherwise 0 volts

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

**Applications**

The analog NCK inputs are used particularly for grinding and laser machines (e.g. for the “analog calipers” NC function).

**Fast analog NCK inputs**

The fast analog inputs must be clock-synchronous. The assignment is defined by MD 10384: HW_CLOCKED_MODULE_MASK.

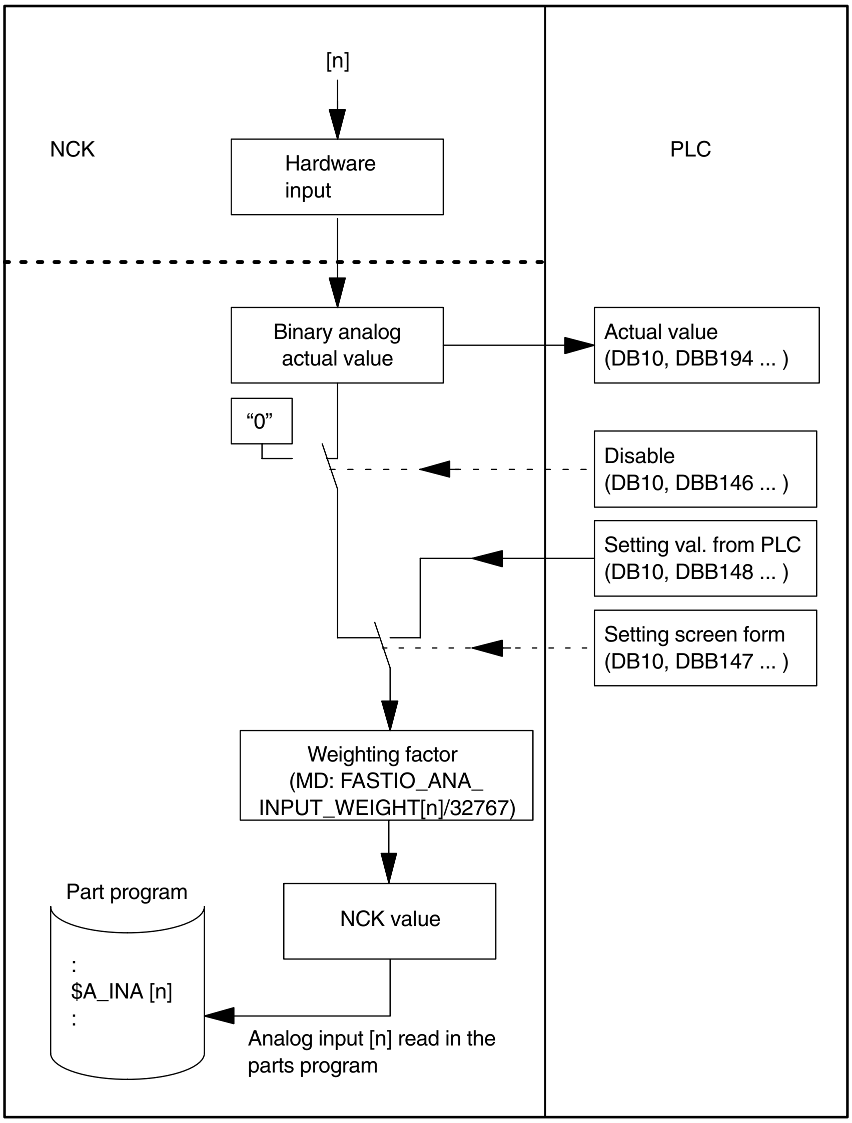



Fig. 2-3 Signal flow for analog NCK inputs

## 2.4.2 Analog outputs of the NCK

**Number**

General MD 10310: FASTIO_ANA_NUM_OUTPUTS (number of active analog NCK outputs) the available analog NCK outputs can be defined.

**Function**

The system variable **$A_OUTA[n]** allows the value at the analog output [n] to be specified directly in the parts program.

Before output to the hardware output, the analog value set by the NCK can be changed by the PLC (see Fig. 2-4).

**Disable output**

The PLC user program is capable of disabling the analog NCK outputs individually with interface signal “Disable analog NCK outputs” (DB10, DBB168). In this case, **0 volts** is output at the analog output (see Fig. 2-4).

**Overwrite screenform**

Every NCK analog value set by the NC parts program can be **overwritten** from the PLC using the overwrite screenform. Previous NCK values are then lost (see Fig. 2-4).

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

1. The output n in question must be preset with the required analog value at the PLC interface “PLC setting for analog output n of the NCK” (DB10, DBB170 to 185).
2. The setting value becomes the new NCK value for the analog output (DB10, DBB166) when the overwrite screenform is activated (signal transition 0 –> 1).

This value remains valid until a new value is set for the NCK by the parts program, for example.

**Setting screen form**

Furthermore, a PLC setting for each output can determine whether the instantaneous (e.g. as specified by NC parts program) or the PLC value specified via the setting screen form (DB10, DBB167) should be sent to the hardware analog output (see Fig. 2-4).

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

1. The output n in question must be preset with the required analog value at the PLC interface “PLC setting for analog output n of the NCK” (DB10, DBB170 to 185).
2. The setting screen form (DB10, DBB167) must be set to “1” for the output in question.

Unlike the overwrite screenform, the current NCK value is not lost when a value is set in the setting screen form. As soon as the PLC sets “0” in the setting screen form, the NCK value is again active.

**Note**

The same setting value (DB10, DBB170 to 185) is used at the PLC interface for the overwrite and the setting screenforms.

**Read setpoint**

The instantaneous NCK value at the analog outputs can be read by the PLC user program (interface signal “setpoint analog output n of NCK” (DB10, DBB210 to 225)). Please note that this setpoint ignores disabling and the setting screen form of the PLC. The setpoint can therefore differ from the real analog value at the hardware output (see Fig. 2-4).

**RESET/end of program**

On end of program or reset, every analog output can be defined as necessary by the PLC user program in the overwrite screenform, setting screen form or disable signal.

**Power ON**

After power ON, the analog outputs are set to “0” in a defined manner. After booting, this can be overwritten in the PLC user program according to the application using the screen forms described above.

**Weighting factor**

Using the weighting factor in the general MD 10330: FASTIO_ANA_OUTPUT_WEIGHT[hw] it is possible to adapt each analog NCK output to the various DACs for programming in the parts program (see Fig. 2-4).

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 the system variable $A_OUTA[n] then places the corresponding voltage value at the analog output in millivolts.

**Binary analog value display**

See Section 2.6

**Exception**

Where the part program contains programmed values for NCK analog outputs that have been defined in MD 10310: FASTIO_ANA_NUM_OUTPUTS, but do not exist as hardware outputs, no alarm is generated. The NCK value can be read by the PLC (IS “Setpoint ...”)

**Application**

This function allows analog outputs to be set instantaneously by bypassing the PLC cycles.

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

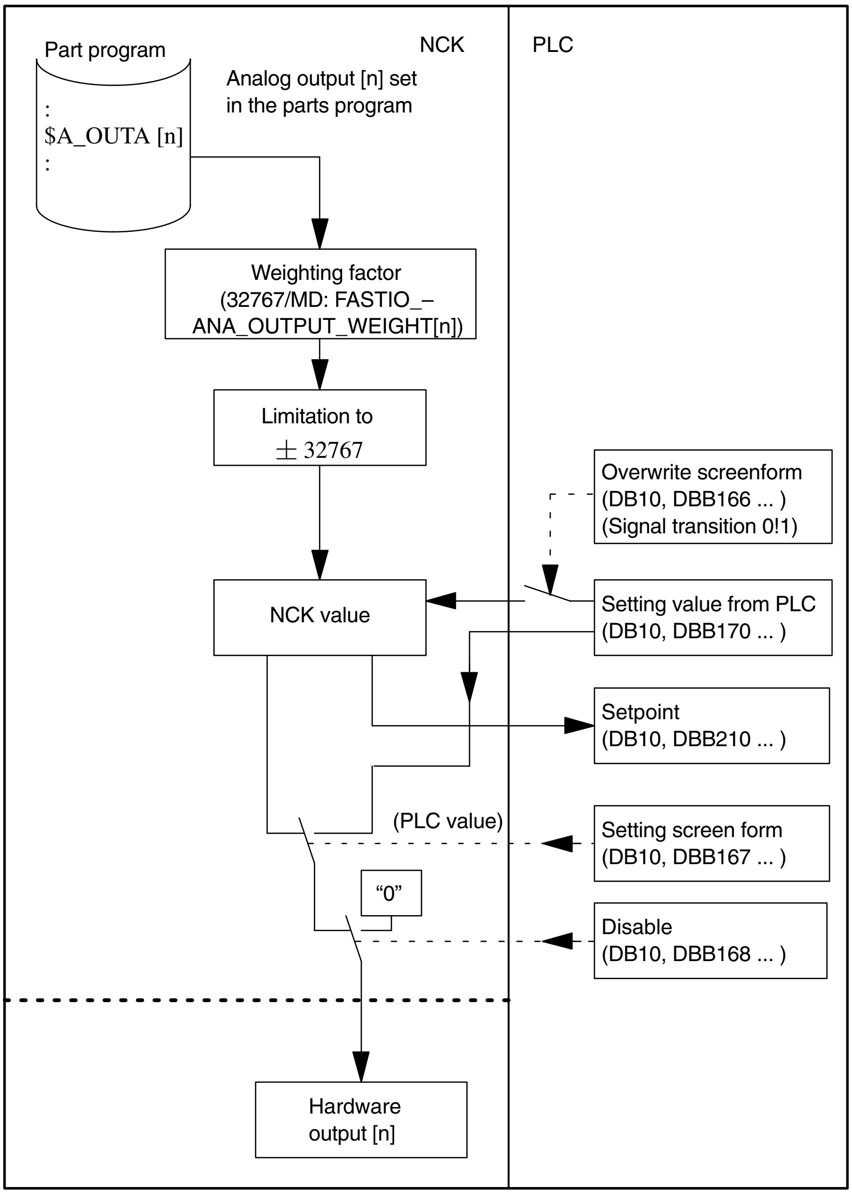



Fig. 2-4 Signal flow for analog NCK outputs

## 2.5 PLC I/Os directly addressable from NC (SW 5 and higher)

**Introduction**

The **high-speed data channel** between the NCK and PLC I/O is processed **directly** and therefore quickly by the PLC operating system.

There is no provision for control of the PLC basic and user programs. Contending access between the NC kernel and the PLC to the same PLC I/O devices is not sensible and can result in faults.

The function is available for

| | |
|---|---|
| SINUMERIK 840D | SW 5.2, SW 6.4 and higher |
| SINUMERIK 810D | SW 3.2, SW 4.4 and higher |

**System variables**

The NC uses **part programs** and **synchronized actions** to access system variables.

For **reading** from PLC:

```
$A_PBB_IN[n]     ; Read byte (8 bits)
$A_PBW_IN[n]     ; Read word (16 bits)
$A_PBD_IN[n]     ; Read Dword (32 bits)
$A_PBR_IN[n]     ; Read real (32 bits float)
```

n Byte offset within the PLC input area

Reading from the parts program causes a preprocessing stop.

For **writing** to PLC:

```
$A_PBB_OUT[n]    ; Write byte (8 bits)
$A_PBW_OUT[n]    ; Write word (16 bits)
$A_PBD_OUT[n]    ; Write Dword (32 bits)
$A_PBR_OUT[n]    ; Write real (32 bits float)
```

n Byte offset within the PLC output area

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

**Variable value ranges**

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

| | |
|---|---|
| $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<br>; (0 ... 4294967295) |
| $A_PBR_OUT[n] | ; (–3.402823466E+38 ... +3.402823466E+38) |

**Transfer times**

The output of values from NCK → PLC (write) takes place at the end of the interpolation cycle if at least one data was written.

Data are read in, as a function of the machine data MD 10398: PLCIO_IN_UPDATE_TIME (SW 6.4 or higher), by sending a request at the end of the interpolation cycle. The new data are available in the subsequent interpolation cycle at the earliest.

Machine data MD 10398: PLCIO_IN_UPDATE_TIME (SW 6.4 or higher) you can set the time within which a request is sent to the PLC. The entered time is set internally to the next higher multiple of an interpolation cycle. If the value of these 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:

MD 10394: PLCIO_NUM_BYTES_IN
Number of PLC I/O input bytes that are read directly by the NC.

MD 10395: PLCIO_LOGIC_ADDRESS_IN
Logical start address of the PLC input I/O from which the data are read.

MD 10396: PLCIO_NUM_BYTES_OUT
Number of PLC I/O output bytes that are written directly by the NC.

MD 10397: PLCIO_LOGIC_ADDRESS_OUT
Logical start address of the PLC output I/O from which the data are written.

MD 10398: PLCIO_IN_UPDATE_TIME (SW 6.4 or higher)
Time within which the data that can be read by means of $A_PBx_IN are updated. The time is set internally to the next higher multiple of the time defined by the interpolation cycle. When 0 is entered (default value), the data are updated in very interpolation cycle.

MD 10399: PLCIO_TYPE_REPRESENTATION (SW6.4 and higher)
Little/Big Endian format representation of $A_PBx_OUT, $A_PBx_IN System variables for PLC I/Os directly controllable by NCK

| | |
|---|---|
| value = 0 | ;(Default) System variables are represented in Little Endian format (i.e. least significant byte at lowest address) |
| value = 1 | ;(PLC standard format, recommended) System variables are represented in Big Endian format (i.e. most significant byte at lowest address) |

The 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**

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

The variables are represented within these areas, depending on the setting of the machine data
MD 10399: PLCIO_TYPE_REPRESENTATION (SW 6.4 or higher) either in little Endian format (=0) or in big Endian format (=1).

Since big Endian format is generally the most common representation form on the PLC (that is, also holds for the PLC I/O), it should generally be used.

**Alignment**

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

SW 5.2 – SW 6.3:

```
$A_PBB_IN[j]     ; j < ([MD 10394: PLCIO_NUM_BYTES_IN])
$A_PBW_IN[j]     ; j:0,2,4,... < ([MD 10394: PLCIO_NUM_BYTES_IN] – 1)
$A_PBD_IN[j]     ; j:0,4,8,... < ([MD 10394: PLCIO_NUM_BYTES_IN] – 3)
$A_PBR_IN[j]     ; j:0,4,8,... < ([MD 10394: PLCIO_NUM_BYTES_IN] – 3)

$A_PBB_OUT[k]  ; k < ([MD 10396: PLCIO_NUM_BYTES_OUT])
$A_PBW_OUT[k]  ; k:0,2,4,... < ([MD 10396: PLCIO_NUM_BYTES_OUT] – 1)
$A_PBD_OUT[k]  ; k:0,4,8,... < ([MD 10396: PLCIO_NUM_BYTES_OUT] – 3)
$A_PBR_OUT[k]  ; k:0,4,8,... < ([MD 10396: PLCIO_NUM_BYTES_OUT] – 3)
```

SW 6.4 and higher:

```
$A_PBB_IN[j]      ; j < ([MD 10394: PLCIO_NUM_BYTES_IN])
$A_PBW_IN[j]      ; j < ([MD 10394: PLCIO_NUM_BYTES_IN] – 1)
$A_PBD_IN[j]      ; j < ([MD 10394: PLCIO_NUM_BYTES_IN] – 3)
$A_PBR_IN[j]      ; j < ([MD 10394: PLCIO_NUM_BYTES_IN] – 3)

$A_PBB_OUT[k]     ; k < ([MD 10396: PLCIO_NUM_BYTES_OUT])
$A_PBW_OUT[k]     ; k < ([MD 10396: PLCIO_NUM_BYTES_OUT] – 1)
$A_PBD_OUT[k]     ; k < ([MD 10396: PLCIO_NUM_BYTES_OUT] – 3)
$A_PBR_OUT[k]     ; k < ([MD 10396: PLCIO_NUM_BYTES_OUT] – 3)
```

Furthermore, the maximum number of bytes available for data exchange must not be exceeded.

## Supplementary conditions

Direct PLC I/Os can be addressed with:

**HW**

Table 2-1 Availability

| NCU HW | Version | PLC SW |
|---|---|---|
| 840D, NCU 561 | NCU 561.2 and higher | from 3.10.13 |
| 840D, NCU 571 | NCU 571.2 and higher | from 3.10.13 |
| 840D, NCU 572 | NCU 572.2 and higher | from 3.10.13 |
| 840D, NCU 573 | NCU 573.2 and higher | from 3.10.13 |
| 810D | CCU2 with PLC315-2 DP | from 3.10.13 |

**SW**

The SINUMERIK SW version must be 5.2 or higher.

**Configuration**

If data are to be read/written over the high-speed data channel, the PLC I/O must

- always be configured as a contiguous block (that is, no gaps between addresses within the block).
- It must be possible for the number of bytes that have to be transferred to be mapped without gaps on the PLC I/O.

**Dynamic response**

The time when the data are read in from the PLC I/O is **not** synchronized with the time when the data are made available to the parts program in the system variables!

**Data transfer (NCK <-> PLC)**

The data buffer is always output complete **to the PLC** I/O, even if only one system variable was assigned within the data buffer.

If values are assigned to several system variables 'simultaneously' (e.g. in order to initialize the PLC I/O), there is no guarantee that they will be transferred in the same interpolation cycle.

## Example for reading

**Reading the PLC I/O with system variable $A_PBx_IN**

The following **assumptions** are made for this example:

- PLC I/O:

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

- $A_PBx_IN is used to read in data from a parts program into R parameters.

- In order to avoid slowing up the PLC user program unnecessarily (OB1), an update time for the internal NCK data buffer (for read access) was configured in machine data $MN_PLCIO_IN_UPDATE_TIME (SW 6.4 or higher) such that an update is only performed every 3rd interpolation cycle.

* The machine data must be set as follows:

  $MN_PLCIO_LOGIC_ADRESS_IN = 420
  ;data are read in from log. addr. 420

  $MN_PLCIO_NUM_BYTES_IN = 11
  ;a total of 11 bytes must be read in

  $MN_PLCIO_IN_UPDATE_TIME = 0.03 ; (SW 6.4 and later)
  ;Update time = 30 msec (interpolation cycle = 12 msec)

  $MN_PLCIO_TYPE_REPRESENTATION = 1 ; (SW 6.4 and later)
  ;data are represented in Big Endian format

* **Booting** of NCK and PLC

  The update (for read access) is now performed in every 3rd interpolation cycle after the NCK and PLC have booted.

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

```
...
R1 = $A_PBW_IN[0]        ;read 16-bit integer
R2 = $A_PBD_IN[2]        ;read 32-bit integer
R3 = $A_PBR_IN[6]        ;read 32-bit float
R4 = $A_PBB_IN[10]       ;read 8-bit integer
...
```

**Example for writing**

**Writing to PLC I/O with $A_PBx_OUT**

The following **assumptions** are made for this example:

* Data are to be output directly to the following PLC I/O:

– 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**.
* The machine data must be set as follows:

  $MN_PLCIO_LOGIC_ADRESS_OUT= 521
  ;data are output from log. addr. 521

  $MN_PLCIO_NUM_BYTES_OUT= 3
  ;a total of 3 bytes must be output

  $MN_PLCIO_TYPE_REPRESENTATION = 1 ; (SW 6.4 and later)
  ;data are represented in Big Endian format

- **Booting** of NCK and PLC

  When the NCK and PLC have booted, cyclic data transfer (for write access) to the PLC I/O does **not** take place.

- **Loading and starting** the part program with the following content:

```
...
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
...
```

## 2.6 Analog value representation of the analog input and output values of the NCK

**Conversion of analog values**

The analog values are only processed by the NCU in a digital form.

Analog input modules convert the analog process signal into a digital value.

Analog output modules convert the digital output value into an analog value.

**Analog value representation**

The digitized analog value is identical for input and output values with the rating range (e.g. voltage range $\pm$ 10V DC).

The analog values are coded in the PLC interface as fixed-point numbers (16 bits including sign) in two's complement (see Table 2-2).

Table 2-2 Digital coding of analog values at the PLC interface

| Resolution | Binary analog value | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | High byte | | | | | | | | Low byte | | | | | | | |
| Bit number | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Significance of the bits | VZ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |

**Sign**

The sign (SG) of the analog value is always in bit 15.

SG is: "0" → +
"1" → –

**Resolution less than 15 bits**

The analog value can be finely adjusted depending on the resolution of the digital/analog converter.

If the resolution of the analog module is less than 15 bits, the analog value is entered left-justified. The free less significant places are filled with zeroes.

Table 2-3 shows how the free bit places are filled with zeroes with a 14-bit and a 12-bit analog value.

With a resolution of 14 bits (including sign), the minimum increment is 1.22mV (10V: 8192). In this case, both less significant bits of the analog value (bit0 and bit1) are always 0.

With a resolution of 12 bits (including sign), the incrementation is 4.8mV (10V: 2048); Bits 0 to 3 are always 0.

Table 2-3 Examples of digital analog value coding

| Resolution | Binary analog value | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | High byte | | | | | | | | Low byte | | | | | | | |
| Bit number | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Significance of the bits | VZ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
| 14-bit analog value | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | **0** | **0** |
| 12-bit analog value | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | **0** | **0** | **0** | **0** |

Details about the resolution and rated range of the analog input and output modules used can be found in:

**References:** /PHD/, SINUMERIK 840D, NCU Manual
/S7H/, SIMATIC S7, Manual

## Examples

Here are two examples of digital analog value coding for a nominal range of 10V and 14-bit resolution.

Example 1

Analog value: 9.5V
Amount (decimal number): 7782 = |9.5(V)|:10(V) * 8192
Amount (binary number): 0111 1001 1001 10
Word (binary number): 0111 1001 1001 10**00**
Word (hexadecimal number): 7998

Example 2

Analog value: –4.12V
Amount (decimal number): 3375 = |–4.12(V)|:10(V) * 8192
Amount (binary number): 0011 0100 1011 11
Two's complement: 1100 1011 0100 01
Word (binary number): 1100 1011 0100 01**00**
Word (hexadecimal number): CB44

## 2.7 Comparator inputs

**Function**

Two internal comparator inputs bytes (with eight comparator inputs each) are available in addition to the digital and analog NCK inputs. The signal status of the comparator inputs is generated on the basis of a comparison between the analog values present at the high-speed analog inputs and high-speed values parameterized in setting data (see Fig. 2-5).

The system variable **$A_INCO[n]** allows the signal status (i.e. the result of the comparison) of comparator input [n] to be scanned directly in the parts program.

The following applies to index n: n = 1 to 8 for comparator byte 1
n = 9 to 16 for comparator byte 2

**Terms**

In this description, the terms “comparator inputs” (with index [n]; range of n: 1 to 8 or 9 to 16) and “comparator input bits” (with index [b]; range of b: 0 to 7) are used.

They are related as follows:

for n = 1 to 8: Comparator input n corresponds to comparator input bit b = n – 1

for n = 9 to 16: Comparator input n corresponds to comparator input bit b = n –9

Example

Comparator input 1 is equivalent to comparator input bit 0.

**Assignment of analog inputs**

General MD 10530: COMPAR_ASSIGN_ANA_INPUT_1 [b] is set to assign an analog input to input bit [b] of comparator byte 1.

Example

MD 10530: COMPAR_ASSIGN_ANA_INPUT_1[0] = 1
MD 10530: COMPAR_ASSIGN_ANA_INPUT_1[1] = 1
MD 10530: COMPAR_ASSIGN_ANA_INPUT_1[7] = 7
Analog input 1 is assigned to input bits 0 and 1 of comparator byte 1
Analog input 7 is assigned to input bit 7 of comparator byte 1

The assignment for comparator byte 2 must be made analogously in the general MD 10531: COMPAR_ASSIGN_ANA_INPUT_2[b].

**Comparator parameterization**

General MD 10540: COMPAR_TYPE_1 is used to set the following parameters for each bit (0 to 7) of comparator byte 1:

- Comparison type screen form (bits 0 to 7)

  The type of comparison conditions is defined for each comparator input bit.

  Bit = 1: Associated comparator input bit is set to “1” if the analog value is $\geq$ the threshold value

  Bit = 0: Associated comparator input bit is set to “0” if the analog value is $\leq$ the threshold value

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

  The comparator bits can also be output directly via the digital NCK outputs in whole bytes. This requires specification in this byte (bits 16 to 23) of the digital NCK output byte to be used (see general MD: 10540 COMPARE_TYPE_1).

- Inversion screen form for outputting the comparator input byte (bits 24 to 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 = 0: The associated comparator input bit is not inverted

  Bit = 1: The 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. For every comparator input bit [b], you must enter a separate threshold value.

MD 41600: COMPAR_THRESHOLD_1[b], threshold values for input bit [b] of comparator byte 1 (b = 0 to 7)

**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. In this case, the byte address for comparator byte 1 (HW byte 128) or 2 (HW byte 129) must be entered in the MD: “Assignment of hardware byte used” associated with the NC function.

Example

NC function “Multiple feedrates in one block”.
Setting in channel-specific MD 21220: MULTFEED_ASSIGN_FASTIN = 129**.**
This activates various feedrate values as a function of the status of comparator byte 2.

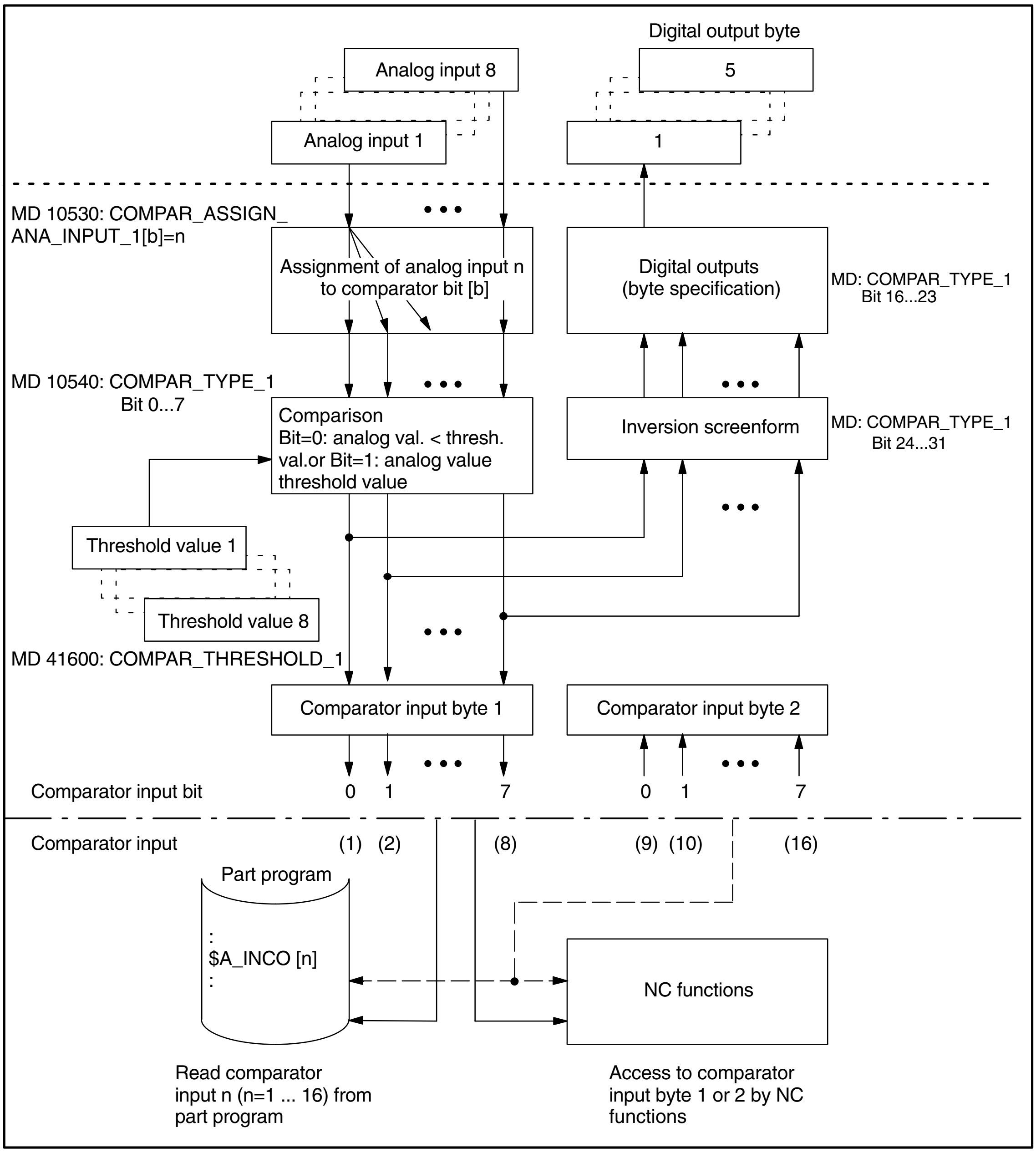



Fig. 2-5 Functional sequence for comparator input byte 1 (or 2)

■

## Notes

# Supplementary Conditions 3

**Availability of the function "Digital and analog NC inputs/outputs"**

Digital and analog CNC inputs/outputs (DI, DO, AI, AO) are available as follows:

- SINUMERIK 840D with NCU 571, SW 2 and higher
  4 DI / 4 DO (on-board)
  32 DI / 32 DO with extension via NCU terminal block
- SINUMERIK 840D with NCU 572/573, SW 2 and higher
  4 DI / 4 DO (on-board)
  32 DI / 32 DO and 8 AI / AO with extension via NCU terminal block

**Analog I/Os for 840Di**

The analog I/Os are connected to the SINUMERK 840Di via PROFIBUS DP.

■

# Notes

# Data Descriptions (MD, SD) 4

## 4.1 General machine data

| | | | | | |
|---|---|---|---|---|---|
| **10300**<br>MD number | **FASTIO_ANA_NUM_INPUTS**<br>Number of active analog NCK inputs | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 8 | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 2.1 | |
| Meaning: | This machine data defines the number of usable analog NCK inputs on the control.<br>Only these analog NCK inputs can be addressed by the NC parts program or assigned by NC functions.<br>If more analog NCK inputs are defined in the machine data than are available in the hardware of the control, the binary analog actual value is set to zero in the control for the inputs that do not exist in the hardware. The NCK value can be altered by the PLC (see Subsection 2.4.1).<br>Note: CPU computing time on the interpolation level is required for processing the digital and analog NCK I/Os. The number of active NCK I/Os should be limited to the demands of the machine so that the interpolation cycle is not overloaded. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **10310**<br>MD number | **FASTIO_ANA_NUM_OUTPUTS**<br>Number of active analog NCK outputs | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 8 | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 2.1 | |
| Meaning: | This machine data defines the number of usable analog NCK outputs on the control.<br>Only these analog NCK outputs can be addressed by the NC parts program or assigned by NC functions.<br>If more analog NCK outputs are defined in the machine data than are available in the hardware of the control, no alarm is triggered. The analog values specified by the parts program can be read by the PLC (see Subsection 2.4.2).<br>Note: CPU computing time on the interpolation level is required for processing the digital and analog NCK I/Os. The number of active NCK I/Os should be limited to the demands of the machine so that the interpolation cycle is not overloaded. | | | | |

| | |
|---|---|
| **10320**<br>MD number | **FASTIO_ANA_INPUT_WEIGHT[hw]**<br>Weighting factor for analog NCK inputs [hw] |
| Default setting: 840D: 10 000<br>FM-NC: 11851 | Minimum input limit: 1 — Maximum input limit: 10 000 000 |
| Changes effective after POWER ON | Protection level: 2 / 4 — Unit: – |
| Data type: DWORD | Applies from SW: 2.1 |
| Meaning: | With this MD a weighting factor can be defined for every analog NCK input [n] with which adaptation to the various A/D converters (depending on the I/O module used; different modules can be used on the FM-NC) is possible.<br>[hw] = Index (0–7) for addressing the external analog inputs<br>The value x must be entered in this machine data which is then to be read in the parts program with the command x = $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.<br>The value read from the AD converter or PLC interface must be multiplied by the factor (FASTIO_ANA_INPUT_WEIGHT / 32767) before it can be read by system variable $A_INA[n] in the parts program (see Fig. 2-3).<br>An internal value of $\pm$ 32767 is generated if the maximum input voltage is applied at the AD converter.<br>Application of the weighting factor for “analog NCK inputs without hardware”: When the weighting factor is set to 32767, the numerical values input from the part program and PLC are identical (1:1 communication between part program and PLC). This is of advantage when the analog NCK inputs/outputs are used purely as PLC inputs/outputs without analog hardware.<br>Note: The comparator threshold values MD 41600: COMPAR_THRESHOLD_1 or MD 41601: COMPAR_THRESHOLD_2 are also scaled to FASTIO_ANA_INPUT_WEIGHT for comparison purposes according to their analog input assignment. |
| Application example(s) | Example 1: Measuring range of analog input module: 0 to 2V (normal range)<br>(FM-NC) maximum value: 2370mV (corresponds to 32767)<br>FASTIO_ANA_INPUT_WEIGHT[0] = 2370<br>In this case, a 2 V analog value is mapped at IS “Actual value ...” (DB10, DBB199...) as digitized value +27648 ($6C00_H$) and the value read in the part program with system variable $A_INA[n] is 2000.<br>Example 2: Measuring range of analog input module: 0 to 10V (normal range)<br>(FM-NC) maximum value: 11.851mV (corresponds to 32767)<br>FASTIO_ANA_INPUT_WEIGHT[1] = 11851<br>An analog value of 10V is the digitized value 27648 (= PLC value); 10000 is read with $A_INA[n]. |
| Related to .... | IS “Setpoint from the PLC for the analog NCK inputs” (DB10, DBB148-163)<br>IS “Setpoint from the PLC for the analog NCK outputs” (DB10, DBB170-185)<br>IS “Setpoint of analog NCK outputs” (DB10, DBB210-225) |

<table>
<tr><td>10330<br>MD number</td><td colspan="5">FASTIO_ANA_OUTPUT_WEIGHT [hw]<br>Weighting factor for analog NCK outputs [hw]</td></tr>
<tr><td colspan="2">Default setting: 840D: 10 000<br>FM-NC: 11852</td><td colspan="2">Minimum input limit: 1</td><td colspan="2">Maximum input limit: 10 000 000</td></tr>
<tr><td colspan="3">Changes effective after POWER ON</td><td colspan="2">Protection level: 2 / 4</td><td>Unit: –</td></tr>
<tr><td colspan="4">Data type: DWORD</td><td colspan="2">Applies from SW: 2.1</td></tr>
<tr><td>Meaning:</td><td colspan="5">With this MD a weighting factor can be defined for every analog NCK output [n] with which adaptation to the various DA converters (depending on the I/O module used) is possible.<br><br>[hw] = Index (0–7) for addressing the external analog outputs<br><br>The value x must be entered in this machine data which is to cause the analog output [n] to be set to the maximum value or set the value +32767 for this output in the PLC interface if $A_OUTA[n] = x is programmed in the parts program. An internal value of<br>$\pm$ 32767 therefore represents the maximum output voltage at the DA converter.<br><br>Application of the weighting factor for “analog NCK outputs without hardware”: When the weighting factor is set to 32767, the numerical values input from the part program and PLC are identical (1:1 communication between part program and PLC). This is of advantage when the analog NCK outputs are used purely as PLC outputs without analog hardware.</td></tr>
<tr><td>Application example(s)</td><td colspan="5">Example (FM-NC):<br><br>Output range of analog output module: 0 to 10V (normal range)<br>maximum value (overrange): 11852mV (corresponds to 32767)<br>FASTIO_ANA_OUTPUT_WEIGHT[0] = 11852<br><br>When $A_OUTA[n] is programmed as 10 000, +27648 ($6C00_H$) is mapped at IS “Setpoint ...“ (DB10, DBB210...); +10V is applied to the analog output.<br><br>Example (840D):<br><br>Output range of the analog output module: 0 to 10V<br>Maximum value: 10 000mV (corresponds to 32767)<br>FASTIO_ANA_OUTPUT_WEIGHT[0] = 10000<br><br>When $A_OUTA[n] is programmed as 10 000, +32767 (i.e. 7FFF) is mapped at IS “Setpoint ...“ (DB10, DBB210...); +10V is generated at the analog output.</td></tr>
<tr><td>Related to ....</td><td colspan="5">IS “Setpoint from the PLC for the analog NCK inputs” (DB10, DBB148-163)<br>IS “Setpoint from the PLC for the analog NCK outputs” (DB10, DBB170-185)<br>IS “Setpoint of analog NCK outputs” (DB10, DBB210-225)</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| **10350**<br>MD number | **FASTIO_DIG_NUM_INPUTS**<br>Number of active digital NCK input bytes | | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 5 | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 2.1 | |
| Meaning: | The number of bytes of the digital NCK inputs that can be used on the control are defined in this machine data.<br><br>These digital NCK inputs can be read directly by the parts program. The signal state at the HW inputs can also be changed by the PLC.<br><br>If more digital NCK inputs are defined in the machine data than are available in the hardware of the control, a signal status of zero is set in the control for the inputs that do not exist in the hardware. The NCK value can be altered by the PLC.<br><br>See Subsection 2.2.1 for a more detailed description. | | | | |
| Application example(s) | Digital NCK inputs 5 to 8 can only be influenced by the PLC (no hardware inputs). | | | | |
| Related to .... | IS “Disable the digital NCK inputs” (DB10, DBB0, DBB122 ...)<br>IS “Set the digital NCK inputs from the PLC” (DB10, DBB1, DBB123 ...)<br>IS “Actual value of the digital NCK inputs” (DB10, DBB60, DBB186 ...) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **10360**<br>MD number | **FASTIO_DIG_NUM_OUTPUTS**<br>Number of active digital NCK output bytes | | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 5 | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 2.1 | |
| Meaning: | The number of bytes of the digital NCK outputs that can be used on the control are defined in this machine data.<br><br>These digital NCK outputs can be set directly by the parts program. The PLC is able to<br>• set the digital outputs with IS “Disable the digital NCK outputs” to “0“ in a defined way.<br>• alter the NCK value with IS “Overwrite screenform for digital NCK outputs”.<br>• specify a PLC value with IS “Setting screenform for digital NCK outputs”.<br><br>If more digital NCK outputs are defined in the machine data than are available in the hardware of the control, no alarm is triggered. The signal states specified by the part program can be read by the PLC.<br><br>See Subsection 2.2.2 for a more detailed description. | | | | |
| Special cases, errors, ...<br>... | Digital NCK outputs 5 to 8 can only be processed by the PLC (no hardware outputs). | | | | |
| Related to .... | IS “Disable the digital NCK outputs” (DB10, DBB4, DBB130 ...)<br>IS “Overwrite screenform of the digital NCK outputs” (DB10, DBB5, DBB131 ...)<br>IS “Setpoint from the PLC for the digital NCK outputs” (DB10, DBB6, DBB132 ...)<br>IS “Setting screen form of the digital NCK outputs” (DB10, DBB7, DBB133 ...)<br>IS “Setpoint of the digital NCK outputs” (DB10, DBB64, DBB190 ...) | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **10361**<br>MD number | **FASTIO_DIG_SHORT_CIRCUIT**<br>Shortcircuits of digital inputs and outputs | | | | | |
| Default setting: 0 | | Minimum input limit: – | | Maximum input limit: – | | |
| Changes effective after POWER ON | | | Protection level: 2/7 | | Unit: – | |
| Data type: DWORD | | | | Applies from SW: 4.2 | | |
| Meaning: | Defined short-circuits between the digital input/output signals of the fast NCK I/Os are achieved by a logic operation between the signals read in from the fast NCK I/Os or PLC interface and the defined output signals. Upon the logic operation, the output signals always remain unchanged, the inputs that are to be accounted for internally are achieved from the read inputs and the logic operation. If more than one output bit is specified for an input bit in overwrite mode, the result is determined by the last assignment defined in the list.<br>The definition for non-existent or non-activated input/outputs is ignored without alarm output.<br>Bits 0–7: Number of input byte to be written (1 – 5)<br>Bits 8–15 Bit number within input byte (1–8)<br>Logic operation:<br>The type of logic is selected by adding a hexadecimal number to the input bit number:<br>00 Overwrite input like output<br>A0 Input is read input ANDed with the status of the specified output<br>B0 Input is read input ORed with the status of the specified output<br>Bits 16–23: Number of output byte (1 – 5)<br>Bits 24–31 Bit number within output byte (1–8) | | | | | |
| Application example(s) | FASTIO_DIG_SHORT_CIRCUIT[0] = H04010302<br>Input 3 of 2nd byte<br>Output 4 of 1st byte (= 4th onboard NCU output)<br>The input state is overwritten by the specified output.<br>FASTIO_DIG_SHORT_CIRCUIT[1] = H0705A201<br>Input 2 of 1st byte (= 2nd onboard NCU input<br>Output 7 of 5th byte<br>The input state is ANDed with the specified output.<br>FASTIO_DIG_SHORT_CIRCUIT[2] = H0103B502<br>Input 5 of 2nd byte<br>Output 1 of 3rd byte<br>The input state is ORed with the specified output. | | | | | |

<table>
<tr><td colspan="2">10362<br>MD number</td><td colspan="4">HW_ASSIGN_ANA_FASTIN[hw]<br>Hardware assignment of external analog NCK inputs</td></tr>
<tr><td colspan="2">Default setting:<br>with 840D/810D: 01000000<br>with 840Di: 050000A0<br>with FM-NC: 02000000</td><td colspan="2">Minimum input limit:<br>with 840D/810D: 01000000<br>with 840Di: 05000000<br>with FM-NC: 02000000</td><td colspan="2">Maximum input limit:<br>with 840D/810D: 011E0802<br>with 840Di: 050003FF<br>with FM-NC: 02070004</td></tr>
<tr><td colspan="2">Changes effective after POWER ON</td><td colspan="2">Protection level: 2 / 4</td><td colspan="2">Unit: Hexadecimal</td></tr>
<tr><td colspan="3">Data type: DWORD</td><td colspan="3">Applies from SW: 2.1</td></tr>
<tr><td>Meaning:</td><td colspan="5">The following 4 bytes define the assignment between the external analog NCK inputs and the hardware<br>Applies to 840D/810D and FM-NC:<br><br>1. byte: I/O No.<br>2nd byte: Submodule No.<br>3rd byte: Module No.<br>4th byte: Segment No.<br><br>Applies to 840Di:<br><br>1. byte: Logical address low<br>2nd byte: Logical address high<br>3rd byte: Not used (00)<br>4th byte: Segment No. for PROFIBUS-DP (05)<br><br>Array length = maximum number of analog inputs on NCK is set in MD10300.<br><br>As soon as value 0 is entered in byte 3 (module no.) external I/Os are no longer processed by the control. A simulated input is defined.<br><br>The hardware assignment is different on the SINUMERIK 840D/810D, 840Di and FM-NC controls.<br><br>The individual bytes are explained under MD 10366: HW_ASSIGN_DIG_FASTIN.<br><br>[hw] = Index (0–7) for addressing the external analog inputs</td></tr>
<tr><td>Related to ....</td><td colspan="5">MD 10366: HW_ASSIGN_DIG_FASTIN<br>MD 10368: HW_ASSIGN_DIG_FASTOUT<br>MD 10364: HW_ASSIGN_ANA_FASTOUT</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| **10364**<br>MD number | **HW_ASSIGN_ANA_FASTOUT[hw]**<br>Hardware assignment of external analog NCK outputs | | | | |
| Default setting:<br>with 840D/810D: 01000000<br>with 840Di: 050000A0<br>with FM-NC: 02000000 | | Minimum input limit:<br>with 840D/810D: 01000000<br>with 840Di: 05000000<br>with FM-NC: 02000000 | | Maximum input limit:<br>with 840D/810D: 011E0802<br>with 840Di: 050003FF<br>with FM-NC: 02070004 | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: Hexadecimal |
| Data type: DWORD | | | | Applies from SW: 2.1 | |
| Meaning: | The following 4 bytes define the assignment between the external analog NCK outputs and the hardware<br>**Applies to 840D/810D and FM-NC:**<br>1. byte: I/O No.<br>2nd byte: Submodule No.<br>3rd byte: Module No.<br>4th byte: Segment No.<br>**Applies to 840Di**:<br>1. byte: Logical address low<br>2nd byte: Logical address high<br>3rd byte: Not used (00)<br>4th byte: Segment No. for PROFIBUS-DP (05)<br>Arry length = maximum number of analog outputs on NCK is set in MD10310.<br>As soon as value 0 is entered in byte 3 (module no.) external I/Os are no longer processed by the control. A simulated input is defined.<br>The hardware assignment is different on the SINUMERIK 840D/810D and FM-NC.<br>The individual bytes are explained under MD 10366: HW_ASSIGN_DIG_FASTIN.<br>[hw] = Index (0–7) for addressing the external analog outputs | | | | |
| Related to .... | MD 10366: HW_ASSIGN_DIG_FASTIN<br>MD 10368: HW_ASSIGN_DIG_FASTOUT<br>MD 10362: HW_ASSIGN_ANA_FASTIN | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **10366**<br>MD number | **HW_ASSIGN_DIG_FASTIN[hw]**<br>Hardware assignment of external digital NCK inputs | | | | |
| Default setting:<br>with 840D/810D: 01000000<br>with 840Di: 05000090<br>with FM-NC: 02000000 | | Minimum input limit:<br>with 840D/810D: 01000000<br>with 840Di: 05000000<br>with FM-NC: 02000000 | | Maximum input limit:<br>with 840D/810D: 011E0802<br>with 840Di: 050003FF<br>with FM-NC: 02070004 | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: Hexadecimal |
| Data type: DWORD | | | | Applies from SW: 2.1 | |

Meaning:

The following 4 bytes define the assignment between the external digital NCK I/Os and the hardware

**Applied to 840D/810D and FM-NC:**

1. byte: I/O No.
2nd byte: Submodule No.
3rd byte: Module No.
4th byte: Segment No.

**Applies to 840Di**:

1. byte: Logical address low
2nd byte: Logical address high
3rd byte: Not used (00)
4th byte: Segment No. for PROFIBUS-DP (05)

Arry length = maximum number of digital input bytes on NCK is set in MD10350.

As soon as value 0 is entered in byte 3 (module no.), the input byte concerned is not processed by the control. A simulated input is defined.

The hardware assignment is different on the SINUMERIK 840D, 840Di and FM-NC.

**840D/810D:**

I/O no.: Number of the I/O byte on the DP compact module (range: 1 to 2; always 1 for analog inputs/outputs)

Submodule no.: Submodule slot on the terminal block into which the DP compact module is slotted (range: 1 to 8)

Module no.: Number of the logical slot into which the terminal block with the external I/Os is slotted.

The assignment of the logical slot to a physical slot is made with MD13010: DRIVE_LOGIC_NR (logical drive number). Each module occupies a physical slot. The first 6 slots on the 810D are permanently assigned.

Segment no.: For 840/810D always 1 (identifier for 611D bus)

Example: HW_ASSIGN_DIGITAL_FASTIN[3] = 01 04 03 02

1. byte: 02 = 2nd input byte of a 16-bit input module
2nd byte: 03 = Input module inserted in slot 3 of terminal block
3rd byte: 04 = Terminal block slotted into logical drive number 4
4th byte: 01 = Identifier for 611D bus

| 10366<br>MD number | **HW_ASSIGN_DIG_FASTIN[hw]**<br>Hardware assignment of external digital NCK inputs |
|---|---|
| Meaning: | **FM-NC:**<br>I/O no.: Number of the input byte on the signal module (range depends on the I/O module: 1 to 4)<br>Submodule no.: Of no relevance for FM-NC (this byte must be preset with 01 in a defined way)<br>Module no.: Number of the physical slot on the local P bus. (Range: 1 to 7)<br>Note: The FM-NC is slot 0; the physical slots for the digital and analog I/Os are located to the left of this.<br>Segment no.: Always 2 for FM-NC (identifier for local P bus)<br>Example: HW_ASSIGN_DIGITAL_FASTIN[3] = 02 04 00 02<br>1. byte: 02 = 2nd input byte of a 16-bit digital input module<br>2nd byte: 00 = Not relevant<br>3rd byte: 04 = Input module slotted into logical drive number 4<br>4th byte: 02 = Identifier for local P bus<br>[hw] = Index (0 to 3) for addressing the external digital input byte |
| Related to .... | MD 10368: HW_ASSIGN_DIG_FASTOUT<br>MD 10362: HW_ASSIGN_ANA_FASTIN<br>MD 10364: HW_ASSIGN_ANA_FASTOUT |

| | | | | | |
|---|---|---|---|---|---|
| **10368**<br>MD number | **HW_ASSIGN_DIG_FASTOUT[hw]**<br>Hardware assignment of external digital NCK outputs | | | | |
| Default setting:<br>with 840D/810D: 01000000<br>with 840Di: 05000090<br>with FM-NC: 02000000 | | Minimum input limit:<br>with 840D/810D: 01000000<br>with 840Di: 05000000<br>with FM-NC: 02000000 | | Maximum input limit:<br>with 840D/810D: 011E0802<br>with 840Di: 050003FF<br>with FM-NC: 02070004 | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: Hexadecimal |
| Data type: DWORD | | | | Applies from SW: 2.1 | |
| Meaning: | The following 4 bytes define the assignment between the external digital NCK outputs and the hardware<br>**Applies to 840D/810D and FM-NC:**<br>1. byte: I/O No.<br>2nd byte: Submodule No.<br>3rd byte: Module No.<br>4th byte: Segment No.<br>**Applies to 840Di**:<br>1. byte: Logical address low<br>2nd byte: Logical address high<br>3rd byte: Not used (00)<br>4th byte: Segment No. for PROFIBUS-DP (05)<br>Arry length = maximum number of digital output bytes on NCK is set in MD10360.<br>As soon as value 0 is entered in byte 3 (module no.), the input byte concerned is not processed by the control. A simulated input is defined.<br>The hardware assignment is different on the SINUMERIK 840D/810D and FM-NC.<br>The individual bytes are explained under MD: HW_ASSIGN_DIG_FASTIN.<br>[hw] = Index (0 to 3) for addressing the external digital output byte | | | | |
| Related to .... | MD 10366: HW_ASSIGN_DIG_FASTIN<br>MD 10362: HW_ASSIGN_ANA_FASTIN<br>MD 10364: HW_ASSIGN_ANA_FASTOUT | | | | |

<table>
<tr><td>10380<br>MD number</td><td colspan="3">HW_UPDATE_RATE_FASTIO [tb]<br>Updating rate of clock-synchronous external NCK I/Os</td></tr>
<tr><td>Default setting: 2</td><td>Minimum input limit: 2</td><td colspan="2">Maximum input limit: 3</td></tr>
<tr><td>Changes effective after POWER ON</td><td>Protection level: 2 / 4</td><td colspan="2">Unit: –</td></tr>
<tr><td>Data type: BYTE</td><td colspan="3">Applies from SW: 2.1</td></tr>
<tr><td>Meaning:</td><td colspan="3">With this machine data, the cycle frequency for the clock-synchronous input and output of the external NCK I/Os is selected (840D only).<br><br>The cycle time applies to all I/O modules on a terminal block that are operated in synchronism with the clock (MD 10384: HW_CLOCKED_MODULE_MASK[tb]=1).<br><br>The selection can be made from the following cycle frequencies:<br><br>Value = 1: Synchronous input/outputs in hardware cycles (not in SW 2)<br>(SYSCLOCK_CYCLE_TIME / SYSCLOCK_SAMPL_TIME_RATIO)<br><br>2: Synchronous input/outputs in position control cycles (default setting)<br>(MD: POSCTR_SYSCLOCK_TIME_RATIO)<br><br>3: Synchronous inputs/outputs in interpolation cycles<br>(MD: IPO_SYSCLOCK_TIME_RATIO)<br><br>Note on index [tb] (tb = 0 to 1):<br>Index [tb] identifies the connected NCU terminal blocks in ascending order of the defined logical module numbers (parameterization with MD: DRIVE_LOGIC_NR “logical drive number”).<br><br>Example: An additional two terminal blocks which are parameterized with the logical drive number 6 and 7 are connected to the drive bus.<br><br>The following assignments are made for the terminal blocks in the control:<br>• HW_UPDATE_RATE_FASTIO[0] parameterizes terminal block 1 with no. 6<br>• HW_UPDATE_RATE_FASTIO[1] parameterizes terminal block 2 with no. 7<br><br>This assignment applies analogously to:<br>MD 10380: HW_UPDATE_RATE_FASTIO[tb] and<br>MD 10384: HW_CLOCKED_MODULE_MASK [tb]<br><br>For more detailed information see<br>References: /FB/, G2, “Velocities, Setpoint/Actual Value Systems, Control”<br><br>Note: Please consider the hardware response times of the external I/O modules used.<br><br>References: /PHD/, SINUMERIK 840D, NCU Manual</td></tr>
<tr><td>Related to ....</td><td colspan="3">MD 10382: HW_LEAD_TIME_FASTIO<br>MD 10384: HW_CLOCKED_MODULE_MASK<br>POSCTR_SYSCLOCK_TIME_RATIO<br>IPO_SYSCLOCK_TIME_RATIO<br>SYSCLOCK_SAMPL_TIME_RATIO<br>DRIVE_LOGIC_NR</td></tr>
<tr><td>References</td><td colspan="3">References: /FB/, G2, “Velocities, Setpoint/Actual Value Systems, Control”</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| **10382**<br>MD number | **HW_LEAD_TIME_FASTIO [tb]**<br>Lead time for clock-synchronous external NCK I/Os | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: ms |
| Data type: DWORD | | | | Applies from SW: 2.1 | |
| Meaning: | A lead time can be defined for digital and analog NCK I/Os (MD 10384: HW_CLOCKED_MODULE_MASK = 1) operated in synchronism with the clock.<br><br>The input signal is stored this length of time before the defined cycle. The output signal is sent to the hardware this same length of time before the defined cycle.<br><br>With NCK inputs, for example, this makes it possible to consider the hardware-determined conversion time of the AD converter so that the digitized analog value is available on the cycle.<br><br>If the value set in this machine data exceeds the fixed cycle time (MD 10380: HW_UPDATE_RATIO_FASTIO), it is limited internally to the largest possible offset (i.e. to the parameterized cycle time).<br><br>The lead time applies to all NCK inputs/outputs of the terminal block addressed with index [tb] which are operated in synchronism with the clock.<br><br>**Note** on index [tb] see MD 10380: HW_UPDATE_RATE_FASTIO. | | | | |
| Related to .... | MD 10380: HW_UPDATE_RATIO_FASTIO<br>MD 10384: HW_CLOCKED_MODULE_MASK | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **10384**<br>MD number | **HW_CLOCKED_MODULE_MASK [tb]**<br>Clock-synchronous processing of external NCK I/Os | | | | |
| Default setting: 00 | | Minimum input limit: 00 | | Maximum input limit: FF | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: Hexadecimal |
| Data type: BYTE | | | | Applies from SW: 2.1 | |
| Meaning: | With SINUMERIK 840D, the I/O modules of the external NCK I/Os can be operated in the following way:<br>• **Asynchronously**, i.e. the input and output values are made available in cycles set by the terminal block which are asynchronous to the internal NC processing cycles.<br>• **Synchronously**, i.e. the input and output values are made available synchronously to a settable internal NC processing cycle.<br><br>The mode of operation can be set via a bit mask (bits 0 to 7) for each individual I/O module of the terminal block addressed with index [tb] (bit 0 for I/O module on slot 1 ... bit 7 for I/O module on slot 8).<br><br>The bits have the following meaning:<br><br>Bit n = 0: I/O module on slot n+1 is operated **asynchronously**<br>Bit n = 1: I/O module on slot n+1 is operated **synchronously**<br><br>The value is of no significance for unassigned slots of the terminal block.<br><br>Example: HW_CLOCKED_MODULE_MASK[0] = 30 (bit mask: 0011 0000)<br>The I/O modules of terminal block 1 on slots 6 and 5 are operated in synchronism with the clock.<br><br>**Note**: Digital NCK inputs/outputs are generally always operated asynchronously. When analog NCK inputs/outputs are used in closed control loops, values often have to be read in and out in synchronism with the clock.<br><br>**Notes**: – On index [tb] see MD 10380: HW_UPDATE_RATE_FASTIO.<br>– Module 6FC5211-0AA10-0AA0 can only operate synchronously. | | | | |
| Related to .... | MD 10382: HW_LEAD_TIME_FASTIO<br>MD 10380: HW_UPDATE_RATIO_FASTIO | | | | |

| **10394**<br>MD number | **PLCIO_NUM_BYTES_IN**<br>Number of direct read inputs bytes of PLC I/Os | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 16 |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: BYTE | Applies from SW: 5.1 | |
| Meaning: | Number of PLC I/O input bytes that can be read directly by the NC.<br>These bytes are transferred via the PLC operating system and not influenced by the PLC basic or user program, resulting in a delay time of less than approximately 0.5ms. They can be read by parts programs/synchronized actions via the following system variables:<br>$A_PBB_IN<br>$A_PBW_IN<br>$A_PBD_IN<br>$A_PBR_IN | |
| Special cases, errors, ... ... | Check and, if necessary, refer to the motor data sheet to correct machine data MD 10394: PLCIO_NUM_BYTES_IN and<br>MD 10395: PLCIO_LOGIC_ADDRESS_IN must be consistent with the PLC configuring data. | |
| Related to .... | MD 10395: PLCIO_LOGIC_ADDRESS_IN | |

| **10395**<br>MD number | **PLCIO_LOGIC_ADDRESS_IN**<br>Start address of direct read input bytes of PLC I/Os | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: plus |
| Changes effective after power ON | Protection level: 2 / 7 | Unit: – |
| Data type: DWORD | Applies from SW: 5.1 | |
| Meaning: | Logical start address of direct PLC input devices<br>From this address onwards, a number of PLCIO_NUM_BYTES_IN bytes for direct use by the NC must be defined by the PLC hardware configuration.<br>These bytes are transferred via the PLC operating system and not influenced by the PLC basic or user program, resulting in a delay time of less than approximately 0.5ms. They can be read by parts programs/synchronized actions via the following system variables:<br>$A_PBB_IN,<br>$A_PBW_IN,<br>$A_PBD_IN,<br>$A_PBR_IN. | |
| Special cases, errors, ... ... | Check and, if necessary, refer to the motor data sheet to correct machine data MD 10394: PLCIO_NUM_BYTES_IN and<br>MD 10395: PLCIO_LOGIC_ADDRESS_IN must be consistent with the PLC configuring data. | |
| Related to .... | MD 10394: PLCIO_NUM_BYTES_IN | |

| **10396**<br>MD number | **PLCIO_NUM_BYTES_OUT**<br>Number of direct write output bytes of PLC I/Os | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 16 |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: BYTE | Applies from SW: 5.1 | |
| Meaning: | Number of PLC I/O output bytes that can be written directly by the NC. These bytes are transferred via the PLC operating system and not influenced by the PLC basic or user program, resulting in a delay time of less than approximately 0.5ms. They can be written/read by parts programs/synchronized actions via the following system variables:<br>$A_PBB_OUT,<br>$A_PBW_OUT,<br>$A_PBD_OUT,<br>$A_PBR_OUT | |
| Special cases, errors, ... ... | Check and, if necessary, refer to the motor data sheet to correct machine data MD 10396: PLCIO_NUM_BYTES_OUT and MD 10397: PLCIO_LOGIC_ADDRESS_OUT must be consistent with the PLC configuring data.<br>In the case of an error, other PLC signals would be overwritten. | |
| Related to .... | MD 10397: PLCIO_LOGIC_ADDRESS_OUT | |

| 10397<br>MD number | PLCIO_LOGIC_ADDRESS_OUT<br>Start address of direct write output bytes of PLC I/Os |
|---|---|
| Default setting: 0 | Minimum input limit: 0 / Maximum input limit: Plus |
| Changes effective after POWER ON | Protection level: 2 / 7 / Unit: – |
| Data type: DWORD | Applies from SW: 5.1 |
| Meaning: | Logical start address of direct PLC output devices From this address onwards, a number of PLCIO_NUM_BYTES_IN bytes must be defined by the PLC hardware configuration.<br>for direct use by the NC.<br>These bytes are transferred via the PLC operating system and not influenced by the PLC basic or user program, resulting in a delay time of less than approximately 0.5ms. They can be written/read by parts programs/synchronized actions via the following system variables:<br>$A_PBB_OUT,<br>$A_PBW_OUT,<br>$A_PBD_OUT,<br>$A_PBR_OUT. |
| Special cases, errors, ... ... | Check and, if necessary, refer to the motor data sheet to correct machine data MD 10396: PLCIO_NUM_BYTES_OUT and MD 10397: PLCIO_LOGIC_ADDRESS_OUT must be consistent with the PLC configuring data. In the case of an error, other PLC signals would be overwritten. |
| Related to .... | MD 10396: PLCIO_NUM_BYTES_OUT |

| 10398<br>MD number | PLCIO_IN_UPDATE_TIME<br>Update time for PLC I/O input cycle |
|---|---|
| Default setting: 0 | Minimum input limit: 0 / Maximum input limit: 10000 |
| Changes effective after POWER ON | Protection level: 2 / 7 / Unit: ms |
| Data type: DOUBLE | Applies from SW: 6.4 |
| Meaning: | Sets the time period within which the PLC I/O data that can be read directly by $A_PBx_IN system variables is updated.<br><br>This time value is rounded up to the next-higher multiple of the period defined by the IPO cycle. |
| Related to .... | MD 10071: IPO_CYCLE_TIME |

<table>
<tr><td>10399<br>MD number</td><td colspan="5">PLCIO_TYPE_REPRESENTATION<br>Little/big endian representation for PLC I/O</td></tr>
<tr><td colspan="2">Default setting: 0</td><td colspan="2">Minimum input limit: 0</td><td colspan="2">Maximum input limit: 1</td></tr>
<tr><td colspan="3">Changes effective after POWER ON</td><td colspan="2">Protection level: 2 / 7</td><td>Unit: –</td></tr>
<tr><td colspan="4">Data type:</td><td colspan="2">Applies from SW: 6.4</td></tr>
<tr><td>Meaning:</td><td colspan="5">Little/big endian format representation of $A_PBx_OUT, $A_PBx_IN system variables for PLC I/Os which can be directly controlled by the NCK.<br><br>value = 0 ;system variables are represented in little endian format<br><br>value = 1 ;system variables are represented in big endian format<br><br>Generally, PLC I/Os must always be controlled in big endian format (value = 1). However, for compatibility reasons, the default setting is little-endian format (value = 0).</td></tr>
</table>

<table>
<tr><td>10530<br>10531<br>MD number</td><td colspan="5">COMPAR_ASSIGN_ANA_INPUT_1[b]<br>COMPAR_ASSIGN_ANA_INPUT_2[b]<br>Hardware assignment of analog inputs for comparator byte 1 (or 2)<br>[bit number]</td></tr>
<tr><td colspan="2">Default setting: 0</td><td colspan="2">Minimum input limit: 0</td><td colspan="2">Maximum input limit: 8</td></tr>
<tr><td colspan="3">Changes effective after POWER ON</td><td colspan="2">Protection level: 2 / 4</td><td>Unit: –</td></tr>
<tr><td colspan="4">Data type: BYTE</td><td colspan="2">Applies from SW: 2.1</td></tr>
<tr><td>Meaning:</td><td colspan="5">With this MD, the analog inputs 1 to 8 are assigned to a bit number of comparator byte 1 or 2. This input bit of the comparator is set to “1“ if the comparison between the applied analog value and the associated threshold value<br>(MD 41600: COMPAR_THRESHOLD_1 or MD 41601: COMPAR_THRESHOLD_2) fulfils the condition parameterized with<br>MD 10540: COMPAR_TYPE_1 or MD 10541: COMPAR_TYPE_2).<br><br>In this case, an analog input can be assigned to a number of comparator input bits.<br><br>The following generally applies for comparator byte 1:<br><br>COMPAR_ASSIGN_ANA_INPUT_1 [b] = n<br><br>with index: b = number of comparator input bit (0 to 7)<br>n = number of analog input (1 to 8)<br><br>Example: COMPAR_ASSIGN_ANA_INPUT_1[0] = 1<br>COMPAR_ASSIGN_ANA_INPUT_1[1] = 2<br>COMPAR_ASSIGN_ANA_INPUT_1[2] = 1<br>COMPAR_ASSIGN_ANA_INPUT_1[3] = 3<br>COMPAR_ASSIGN_ANA_INPUT_1[4] = 3<br>COMPAR_ASSIGN_ANA_INPUT_1[5] = 1<br>COMPAR_ASSIGN_ANA_INPUT_1[6] = 1<br>COMPAR_ASSIGN_ANA_INPUT_1[7] = 1<br>Analog input 1 affects input bit 0, 2 , 5, 6 and 7 of comparator byte 1<br>Analog input 2 affects input bit 1 of comparator byte 1<br>Analog input 3 affects input bits 3 and 4 of comparator byte 1<br><br>The same also applies to comparator byte 2 with respect to COMPAR_ASSIGN_ANA_INPUT_2[b].<br><br>See Section 2.7 for a more detailed description.</td></tr>
<tr><td>Related to ....</td><td colspan="5">MD 10540: COMPAR_TYPE_1<br>MD 10541: COMPAR_TYPE_2</td></tr>
</table>

| 10540<br>10541<br>MD number | COMPAR_TYPE_1<br>COMPAR_TYPE_2<br>Parameterization for comparator byte 1 or 2 | |
|---|---|---|
| Default setting: 0 | Minimum input limit: | Maximum input limit: |
| Changes effective after POWER ON | Protection level: 2 / 4 | Unit: Binary mask |
| Data type: DWORD | Applies from SW: 2.1 | |
| Meaning: | The following parameters can be set with COMPAR_TYPE_1 for the individual input bits (0 to 7) of comparator byte 1:<br>• Bits 0 to 7: Type of comparison mask (for comparator input bit 0 to 7)<br>Bit = 1: Input bit = 1 when analog value ≥ threshold value<br>Bit = 0: Input bit = 1 when analog value < threshold value<br>(threshold value set with MD 41600: COMPAR_THRESHOLD_1 or MD 41601: COMPAR_THRESHOLD_2)<br>• Bits 8 to 15: Not assigned (to be set to 0 in a defined way)<br>• Bits 16 to 23: Assignment of a hardware output byte for the output of the comparator states (stating byte address)<br>Byte = 0: No output via digital NCK outputs<br>Byte = 1: Output via digital on-board NCK outputs (1 to 4)<br>Byte = 2: Output via external digital NCK outputs 9 to 16<br>Byte = 3: Output via external digital NCK outputs 17 to 24<br>Byte = 4: Output via external digital NCK outputs 25 to 32<br>Byte = 5: Output via external digital NCK outputs 33 to 40<br>• Bits 24 to 31: Inversion screen form for the output of comparator states (bit 0 to 7)<br>Bit = 0: Input bit is not inverted<br>Bit = 1: Input bit is inverted<br><br>The same also applies to comparator byte 2 with COMPAR_TYPE_2.<br><br>For more information please refer to Section 2.7. | |
| Related to .... | MD 10530: COMPAR_ASSIGN_ANA_INPUT_1<br>MD 10531: COMPAR_ASSIGN_ANA_INPUT_2<br>MD 41600: COMPAR_THRESHOLD_1<br>MD 41601: COMPAR_THRESHOLD_2<br>MD 10360: FASTIO_DIG_NUM_OUTPUTS | |

## 4.2 General setting data

| 41600<br>41601<br>MD number | COMPAR_THRESHOLD_1[b]<br>COMPAR_THRESHOLD_2[b]<br>Threshold values for comparator byte 1 or 2 [bit 0 to 7] | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: ± 10 000 |
| Changes effective immediately | Protection level: 7 | Unit: mV |
| Data type: DOUBLE | Applies from SW: 2.1 | |
| Meaning: | COMPAR_THRESHOLD_1[b] defines the threshold values for the individual input bits[b] of comparator byte 1.<br><br>The same also applies to comparator byte 2 with COMPAR_THRESHOLD_2[b].<br><br>Index [b]: Bits 0 – 7<br><br>See Section 2.7 for a more detailed description. | |
| Related to .... | MD 10530: COMPAR_ASSIGN_ANA_INPUT_1<br>MD 10531: COMPAR_ASSIGN_ANA_INPUT_2<br>MD 10540: COMPAR_TYPE_1<br>MD 10541: COMPAR_TYPE_2 | |

■

# Signal Descriptions 5

## 5.1 NC specific signals

The overview in Subsections 5.1.1 and 5.1.3 lists only the signals which are described below. For a complete list of signals, please see

**References:** /LIS/, Lists

### 5.1.1 Overview of signals from PLC to NC (DB10)

| 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<br>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 digital NCK inputs on the PLC<br>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<br>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<br>Digital outputs without hardware #) | | | | On-board outputs §) | | | |
| | Output 8 | Output 7 | Output 6 | Output 5 | Output 4 | Output 3 | Output 2 | Output 1 |
| **6** | Setting value from PLC for the digital NCK outputs<br>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<br>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 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.

§) With the 840D, the NCK digital inputs and outputs 1 to 4 are provided as onboard hardware inputs and outputs. No hardware I/Os are available for bits 0 to 3 on the FM-NC. These can be processed by the PLC according to #).

| 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 digital NCK inputs on the PLC | | | | | | | |
| | 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 digital NCK inputs on the PLC | | | | | | | |
| | 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 digital NCK inputs on the PLC | | | | | | | |
| | 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 digital NCK inputs on the PLC | | | | | | | |
| | 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** | Setting value from PLC for 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** | Setting value from PLC for 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** | Setting value from PLC for 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 |

| DB10 | Signals to NC interface PLC → NC | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **DBB** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **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** | Setting value from PLC for 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 | | | | | | | |

### 5.1.2 Description of signals from PLC to NC (DB10)

| | | |
|---|---|---|
| **DB10**<br>**DBB0, 122, 124, 126, 128**<br>Data Block | **Disable digital NCK inputs**<br><br>Signal(s) to NC (PLC → NC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 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 signal transition 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 parts program.<br><br>See Subsection for more detailed information 2.2.1 | |
| Related to .... | IS “Setting by PLC of digital NCK inputs” (DB10, DBB1)<br>IS “Actual value of digital NCK inputs” (DB10, DBB60)<br>MD 10350: FASTIO_DIG_NUM_INPUTS | |

| | | |
|---|---|---|
| **DB10**<br>**DB1, 123, 125, 127, 129**<br>Data Block | **Setting by PLC of digital NCK inputs**<br><br>Signal(s) to NC (PLC → NC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 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 signal transition 1 ——> 0 | The signal state at the NCK input is enabled for read access by the NC parts program. However, the state can be accessed only if the NCK input is not disabled by the PLC (IS “Disable digital NCK inputs” = 0).<br><br>See Subsection for more detailed information 2.2.1 | |
| Related to .... | IS “Disable digital NCK inputs” (DB10, DBB0)<br>IS “Actual value for digital NCK inputs” (DB10, DBB60)<br>MD 10350: FASTIO_DIG_NUM_INPUTS | |

| | | |
|---|---|---|
| **DB10**<br>**DBB4, 130, 134, 138, 142**<br>Data Block | **Disable digital NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | The digital NCK output is disabled. “0V” is output in a defined way at the hardware output. | |
| Signal state 0 or signal transition 1 ——> 0 | The digital output of the NCK is enabled. As a result, the value set by the NC parts program or the PLC is output at the hardware output.<br><br>See Subsection for more detailed information 2.2.2 | |
| Related to .... | IS “Overwrite screenform for the digital NCK outputs” (DB10, DBB5)<br>IS “Setting screen form for the digital NCK outputs” (DB10, DBB7)<br>IS “Setting by PLC for the digital NCK outputs” (DB10, DBB6)<br>MD 10360: FASTIO_DIG_NUM_OUTPUTS | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB5, 131, 135, 139, 143**<br>Data Block | **Overwrite screenform for digital NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | | |
| Edge evaluation: no | Signal(s) updated: Cyclic | | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | On signal transition 0 –> 1 the previous NCK value is overwritten by the setting value (IS “Setting value from PLC for digital NCK outputs”). The previous NCK value, which, for example, was directly set by the parts program, is lost.<br><br>The signal status defined by the setting value forms the new NCK value.<br><br>For more information please see Subsection 2.2.2 | | |
| Signal state 0 or signal transition 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 screenform (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. | | |
| Related to .... | IS “Disable digital NCK outputs” (DB10, DBB4)<br>IS “Setting screen form of the digital NCK outputs” (DB10, DBB7)<br>IS “Setting value from PLC for digital NCK outputs” (DB10, DBB6)<br>MD 10360: FASTIO_DIG_NUM_OUTPUTS | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB6, 132, 136, 140, 144**<br>Data Block | **Setting by PLC of digital NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | | |
| Edge evaluation: no | Signal(s) updated: Cyclic | | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The signal status for the digital hardware output can be changed by the PLC with the setting value. There are two possibilities:<br><br>1. With the ’overwrite screenform’:<br>On signal transition 0 –> 1 the PLC overwrites the previous ’NCK value’ with the ’setting value’. This is the new ’NCK value’.<br><br>2. With the ’setting screen form’:<br>On signal state 1, the ’PLC value’ is activated. The value used is the ’setting value’.<br><br>On setting value “1”, signal level 1 is output at the hardware output; on “0”, 0 signal level is output. The corresponding voltage values are given in<br><br>**References:** /PHD/, SINUMERIK 840D, NCU Manual<br><br>For more information please see Subsection 2.2.2 | | |
| Signal state 0 or signal transition 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 screenform (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. | | |
| Related to .... | IS “Disable digital NCK outputs” (DB10, DBB4)<br>IS “Overwrite screenform of the digital NCK outputs” (DB10, DBB5)<br>IS “Setting screen form of the digital NCK outputs” (DB10, DBB7)<br>MD 10360: FASTIO_DIG_NUM_OUTPUTS | | |

| DB10<br>**DBB7, 133, 137, 141, 145**<br>Data Block | **Setting screen form for digital NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 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 “Setting value from PLC for digital NCK outputs”.<br>The current NCK value is not lost.<br>For more information please see Subsection 2.2.2 | |
| Signal state 0 or signal transition 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 screenform (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. | |
| Related to .... | IS “Disable digital NCK outputs” (DB10, DBB4)<br>IS “Overwrite screenform of the digital NCK outputs” (DB10, DBB5)<br>IS “Setting value from PLC for the digital NCK outputs” (DB10, DBB6)<br>MD 10360: FASTIO_DIG_NUM_OUTPUTS | |

| **DB10**<br>**DBB146**<br>Data Block | **Disable analog NCK inputs**<br><br>Signal(s) to NC (PLC → NC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 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 signal transition 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 parts program if the setting screen form is set to 0 signal by the PLC for this NCK input.<br>See Subsection for more detailed information 2.4.1 | |
| Related to .... | IS “Setting screen form of the analog NCK inputs” (DB10, DBB147)<br>IS “Setting value from PLC for the analog NCK inputs” (DB10, DBB148)<br>IS “Actual value of the analog NCK inputs” (DB10, DBB199 ...)<br>MD 10300: FASTIO_ANA_NUM_INPUTS | |

| **DB10**<br>**DBB147**<br>Data Block | **Setting screen form of analog NCK inputs**<br><br>Signal(s) to NC (PLC → NC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The setting value from the PLC acts as the enabled analog value<br>(IS “Setting value from PLC for analog NCK inputs”). | |
| Signal state 0 or signal transition 1 ––> 0 | The analog value at the NCK input is enabled for read access by the NC parts program on condition that the NCK input is not disabled by the PLC (IS “Disable analog NCK inputs” = 0).<br>See Subsection for more detailed information 2.4.1 | |
| Related to .... | IS “Disable analog NCK inputs” (DB10, DBB146)<br>IS “Setting value from PLC for the analog NCK inputs” (DB10, DBB148 to 163)<br>IS “Actual value of analog NCK inputs” (DB10, DBB199 to 209)<br>MD 10300: FASTIO_ANA_NUM_INPUTS | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB148 –163**<br>Data Block | **Setting value from PLC for analog NCK inputs**<br><br>Signal(s) to NC (PLC → NC) | | |
| Edge evaluation: no | Signal(s) updated: Cyclic | | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 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.<br><br>The setting value from the PLC becomes active as soon as IS “Setting screen form” is set to “1“.<br><br>The setting value from the PLC is specified as a fixed point number (16 bit value including sign) in 2’s complement (see Section 2.6).<br><br>For more information please see Subsection 2.4.1 | | |
| Related to .... | IS “Disable analog NCK inputs” (DB10, DBB146)<br>IS “Setting screen form of analog NCK inputs” (DB10, DBB147)<br>IS “Actual value of analog NCK inputs” (DB10, DBB199 to 209)<br>MD 10300: FASTIO_ANA_NUM_INPUTS | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB166**<br>Data Block | **Overwrite screenform of analog NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | | |
| Edge evaluation: no | Signal(s) updated: Cyclic | | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 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.<br><br>The analog value specified by the PLC setting value forms the new NCK value.<br><br>For more information please see Subsection 2.4.2 | | |
| Signal state 0 or signal transition 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 screenform (on signal transition 0 –> 1) and the setting screen form (signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program. | | |
| Related to .... | IS “Disable analog NCK outputs” (DB10, DBB168)<br>IS “Setting screen form of analog NCK outputs” (DB10, DBB167)<br>IS “Setting value from PLC for the analog NCK outputs” (DB10, DBB170–185)<br>MD 10310: FASTIO_ANA_NUM_OUTPUTS | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB167**<br>Data Block | **Setting screen form of analog NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | | |
| Edge evaluation: no | Signal(s) updated: Cyclic | | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 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”.<br><br>The current NCK value is not lost.<br><br>For more information please see Subsection 2.4.2 | | |
| Signal state 0 or signal transition 1 ––> 0 | The NCK value is output at the analog hardware output. | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB167**<br>Data Block | **Setting screen form of analog NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | | |
| Special cases, errors, ......| Note: The PLC interface for the setting value is used both by the overwrite screen-form (on signal transition 0 –> 1) and the setting screen form (signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program. | | |
| Related to .... | IS “Disable analog NCK outputs” (DB10, DBB168)<br>IS “Overwrite screenform of analog NCK outputs” (DB10, DBB166)<br>IS “Setting value from PLC of the analog NCK outputs” (DB10, DBB170–185)<br>MD 10310: FASTIO_ANA_NUM_OUTPUTS | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB168**<br>Data Block | **Disable analog NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 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 signal transition 1 ——> 0 | The analog output of the NCK is enabled. As a result, the value set by the NC parts program or the PLC is output at the hardware output.<br><br>For more information please see Subsection 2.4.2 | | |
| Related to .... | IS “Overwrite screenform of analog NCK outputs” (DB10, DBB166)<br>IS “Setting screen form of analog NCK outputs” (DB10, DBB167)<br>IS “Setting value from PLC of the analog NCK outputs” (DB10, DBB170-185)<br>MD 10310: FASTIO_ANA_NUM_OUTPUTS | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB170 –185**<br>Data Block | **Setting value from PLC for analog NCK outputs**<br><br>Signal(s) to NC (PLC → NC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ——> 1 | With this setting value, the value for the analog hardware output can be changed by the PLC. There are two possibilities:<br><br>1. With the ’overwrite screenform’:<br>On signal transition 0 –> 1 the PLC overwrites the previous ’NCK value’ with the ’setting value’. This is the new ’NCK value’.<br><br>2. With the ’setting screen form’:<br>On signal state 1, the ’PLC value’ is activated. The value used is the ’setting value’.<br><br>The setting value from the PLC is specified as a fixed point number (16 bit value including sign) in 2’s complement.<br><br>For more information please see Subsection 2.4.2 | | |
| Signal state 0 or signal transition 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 (on signal transition 0 –> 1) and the setting screen form (signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program. | | |
| Related to .... | IS “Disable analog NCK outputs” (DB10, DBB168)<br>IS “Overwrite screenform of analog NCK outputs” (DB10, DBB166)<br>IS “Setting screen form of analog NCK outputs” (DB10, DBB167)<br>MD 10310: FASTIO_ANA_NUM_OUTPUTS | | |

## 5.1.3 Overview of signals from NC to PLC (DB10)

| DB10 | Signals to 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.

§) With the 840D, the NCK digital inputs and outputs 1 to 4 are provided as onboard hardware inputs and outputs. No hardware I/Os are available for bits 0 to 3 on the FM-NC. These can be processed by the PLC according to #).

| DB10 | Signals to 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 |

## 5.1.4 Description of signals from NC to PLC (DB10)

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB60,**<br>**186–189**<br>Data Block | **Actual value for digital NCK inputs**<br><br><br>Signal(s) to PLC (NC → PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | Signal level "1" is active at the digital hardware input of the NCK. | | |
| Signal state 0 or signal transition 1 ––> 0 | Signal level "0" is active at the digital hardware input of the NCK.<br>For more information please see Subsection 2.2.1 | | |
| Special cases, errors, ... ... | The influence of IS "Disable digital NCK inputs" is ignored for the actual value. | | |
| Related to .... | IS "Disable digital NCK inputs" (DB10, DBB0)<br>MD 10350: FASTIO_DIG_NUM_INPUTS | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB64,**<br>**190 –193**<br>Data Block | **Setpoint for digital NCK outputs**<br><br><br>Signal(s) to PLC (NC → PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The NCK value for the digital output currently set (setpoint) is "1". | | |
| Signal state 0 or signal transition 1 ––> 0 | The NCK value for the digital output currently set (setpoint) is "0".<br>See Subsection for more detailed information 2.2.2 | | |
| Signal irrelevant for ... ... | This 'setpoint' is only output to the hardware output under the following conditions:<br>1. Output is not disabled (IS "Disable digital NCK outputs")<br>2. PLC has switched to the NCK value (IS "Setting screen form for digital NCK inputs")<br>As soon as these conditions are fulfilled, the setpoint of the digital output corresponds to the 'actual value'. | | |
| Related to .... | IS "Disable digital NCK outputs" (DB10, DBB4)<br>IS "Overwrite screenform of the digital NCK outputs" (DB10, DBB5)<br>IS "Setting value from PLC for the digital NCK outputs" (DB10, DBB6)<br>IS "Setting screen form of digital NCK outputs" (DB10, DBB7)<br>MD 10310: FASTIO_DIG_NUM_OUTPUTS | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB194 –209**<br>Data Block | **Actual value for analog NCK inputs**<br><br>Signal(s) to PLC (NC → PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ——> 1 | The analog value applied to the analog NCK input is signalled to the PLC.<br>The actual value is set as a fixed point number (16 bit value including sign) in 2's complement by the NCK (see Section 2.6).<br>For more information please see Subsection 2.4.1 | | |
| Signal state 0 or signal transition 1 ——> 0 | The effect of the PLC on the analog value (e.g. with IS "Disable analog NCK inputs") is ignored. | | |
| Related to .... | IS "Disable analog NCK inputs" (DB10, DBB146)<br>IS "Setting screen form of analog NCK inputs" (DB10, DBB147)<br>IS "Setting value from PLC for the analog NCK inputs" (DB10, DBB148 to 163)<br>MD 10300: FASTIO_ANA_NUM_INPUTS | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB210 –225**<br>Data Block | **Setpoint for analog NCK outputs**<br><br>Signal(s) to PLC (NC → PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ——> 1 | The current set NCK value for the analog output (setpoint) is signalled to the PLC.<br>The setpoint is set as a fixed point number (16 bit value including sign) in 2's complement by the NCK (see Section 2.6).<br>For more information please see Subsection 2.4.2 | | |
| Signal state 0 or signal transition 1 ——> 0 | This "setpoint" is only output at the hardware output when the following conditions are fulfilled:<br>1. Output is not disabled (IS "Disable analog NCK outputs")<br>2. The PLC has switched to the NCK value (IS "Setting screen form of analog NCK outputs") | | |
| Related to .... | IS "Disable analog NCK outputs" (DB10, DBB168)<br>IS "Overwrite screenform for analog NCK outputs" (DB10, DBB166)<br>IS "Setting value by PLC of the analog NCK outputs" (DB10, DBB170-185)<br>IS "Setting screen form of analog NCK outputs" (DB10, DBB167)<br>MD 10310: FASTIO_ANA_NUM_OUTPUTS | | |

■

# Notes

# Example 6

None

# Data Fields, Lists 7

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **General Signals from NC to PLC** | | | |
| 10 | 0, 122, 124, 126, 128 | Disable digital NCK inputs | |
| 10 | 1, 123, 125, 127, 129 | Setting digital NCK inputs on the PLC | |
| 10 | 4, 130, 134, 138, 142 | Disable digital NCK outputs | |
| 10 | 5, 131, 135, 139, 143 | Overwrite screenform for digital NCK outputs | |
| 10 | 6, 132, 136, 140, 144 | Setting value from PLC for the digital NCK outputs | |
| 10 | 7, 133, 137, 141, 145 | Setting screen form for digital NCK outputs | |
| 10 | 146 | Disable analog NCK inputs | |
| 10 | 147 | Setting screen form for analog NCK inputs | |
| 10 | 148–163 | Setting value from PLC for the analog NCK inputs | |
| 10 | 166 | Overwrite screenform for analog NCK outputs | |
| 10 | 167 | Setting screen form for analog NCK outputs | |
| 10 | 168 | Disable analog NCK outputs | |
| 10 | 170–185 | Setting value from PLC for the analog NCK outputs | |
| **Signals from NC to PLC** | | | |
| 10 | 60, 186–189 | Actual value for digital NCK inputs | |
| 10 | 64, 190–193 | Setpoint for digital NCK outputs | |
| 10 | 194–209 | Actual value for analog NCK inputs | |
| 10 | 210–225 | Setpoint for analog NCK outputs | |

# 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ... )** | | | |
| 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 | |
| 10380 | HW_UPDATE_RATE_FASTIO | Updating rate of clock-synchronous external NCK I/Os | |
| 10382 | HW_LEAD_TIME_FASTIO | Lead time for clock-synchronous external NCK I/Os | |
| 10384 | HW_CLOCKED_MODULE_MASK | Processing of external NCK I/Os in synchronism with the clock | |
| 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 | |
| 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 | |
| **Channelspecific ($MC_ ... )** | | | |
| 21220 | MULTFEED_ASSIGN_FASTIN | Assignment of input bytes of NCK I/Os for “Multiple feedrates in one block” | V1 |

## 7.3 Setting data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($SN_ ...)** | | | |
| 41600 | COMPAR_THRESHOLD_1 | Threshold values for comparator byte 1 | |
| 41601 | COMPAR_THRESHOLD_2 | Threshold values for comparator byte 2 | |

## 7.4 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, Diagnostics Guide
and the online help of MMC 101/102/103 systems.

■

# Notes

# SINUMERIK 840D/840Di/810D Extended Functions Description of Functions (FB2)

# Several Operator Panel Fronts and NCUs (B3)

■

# 1 Brief Description

## 1.1 Topology of distributed system configurations

**Features**

Rotary indexing machines, multi-spindle turning machines and complexNC production centers all have one or some of the following features:

- More than one NCU owing to large number of axes and channels
- Large dimensions and spatial separation necessitate several operating units (MMC operator panel fronts and MCP machine control panels or HT6 handheld terminal)
- Modular machine concept, e.g. through distributed control cubicles

SW 5 offers a flexible general solution for these applications.

**Software version**

The software versions specified in this documentation refer to the SINUMERIK 840D control; the parallel software version for the SINUMERIK 810D control (if the function is released, see /BU/, Catalog NC 60.1) is not specified explicitly.
The following applies:

Table 1-1 Equivalent software version

| SINUMERIK 840D software version | | SINUMERIK 810D software version |
|---|---|---|
| 4.1 (12.98) | equivalent to | 3.1 (12.98) |
| 4.3 (12.97) | equivalent to | 2.3 (12.97) |
| 3.7 (03.97) | equivalent to | 1.7 (03.97) |
| 5.1 (12.98) | equivalent to | 3.1 (12.98) |
| 5.2 (08.99) | equivalent to | 3.2 (08.99) |
| 5.3 (04.00) | equivalent to | 3.3 (04.00) |

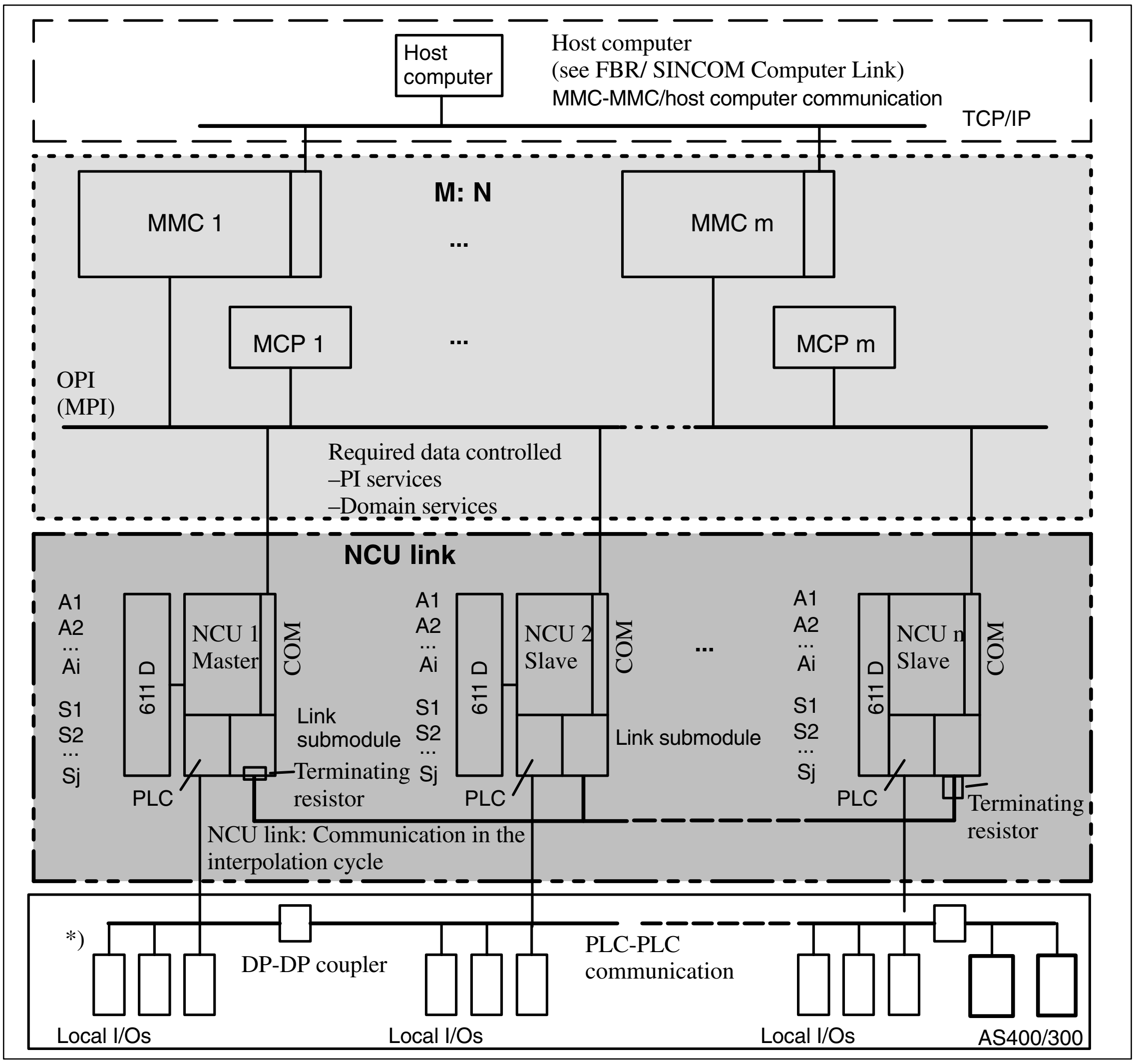



Fig. 1-1 Topology of distributed system configurations

*) The term PLC-PLC communication refers either to
– PLC-PLC cross-communication master, slave C.) or
– PLC local I/Os.

The two areas highlighted in the topology display above identify two communications functions to be examined separately in terms of configuration and utilization.

**M: N**
**(m MMC : n NCU)**

Assignment of several MMC, MCP or HT6 control units (M) to several NCUs (N).

- Bus addresses, bus type
- Properties of the MMC/HT6
  – Main control panel/secondary control panel
- Dynamic switchover from MMCs/MCPs or HT6s to other NCUs

For a detailed description of these operations, please refer to Chapter 2.

Actions are required for the use of M:N during:

- Hardware planning (see /PHD/ )
- Parameterization in files (see Chapter 2)
- PLC programming (see /FB1/, P3)
- Operation (see /BAD/, /BEM/ )

For applications/configurations matching the examples described in Chapter 6, the notation examples can be copied directly or modified slightly. The aspects involved in file parameterization, PLC programming and operation are specified in the quick installation guide.

For different applications, please refer to the full description in Chapter 2 and the source documents specified above.

**NCU link**

The functions for the NCU link are based on additional communication between NCUs in the interpolation cycle. The NCU link allows:

- Subordination of a physical axis to several different NCUs
- Interpolation across the NCUs
- An increase in the number of usable axes for an NCU grouping
- An increase in the number of channels for an NCU grouping
- Provision of axis data and signals on the NCU to which a non-local axis is temporarily assigned
- User communication via the NCU grouping by means of link variables

For more information about this topic, please refer to Chapters 1 and 2.

**Lead link axes as of SW 6**

In software version 6 and higher, following axes can be traversed by an NCU if the associated leading axis is being traversed by another NCU. The NCU link communication handles the necessary exchange of axis data. See 1.4.

**NCU link with different IPO cycles SW 6**

For special applications such as eccentric turning, software version 6 and higher provides support for an NCU link between NCUs with different interpolation cycles. For details please refer to 1.4.6 and 2.11.

**Host computer**

Communication between host computers and control units is described in
**References:** /FBR/, SINCOM Computer Link

**PLC-PLC communication**

DP Master, DP Slave, DP-DP coupler, cross-communication via PBK

**Bus capacities**

The buses illustrated in the above topology diagram are specially designed for their transmission tasks. The resultant communication specifications are shown in the next figure:

– Number of bus nodes
– Baud rate
– Synchronization

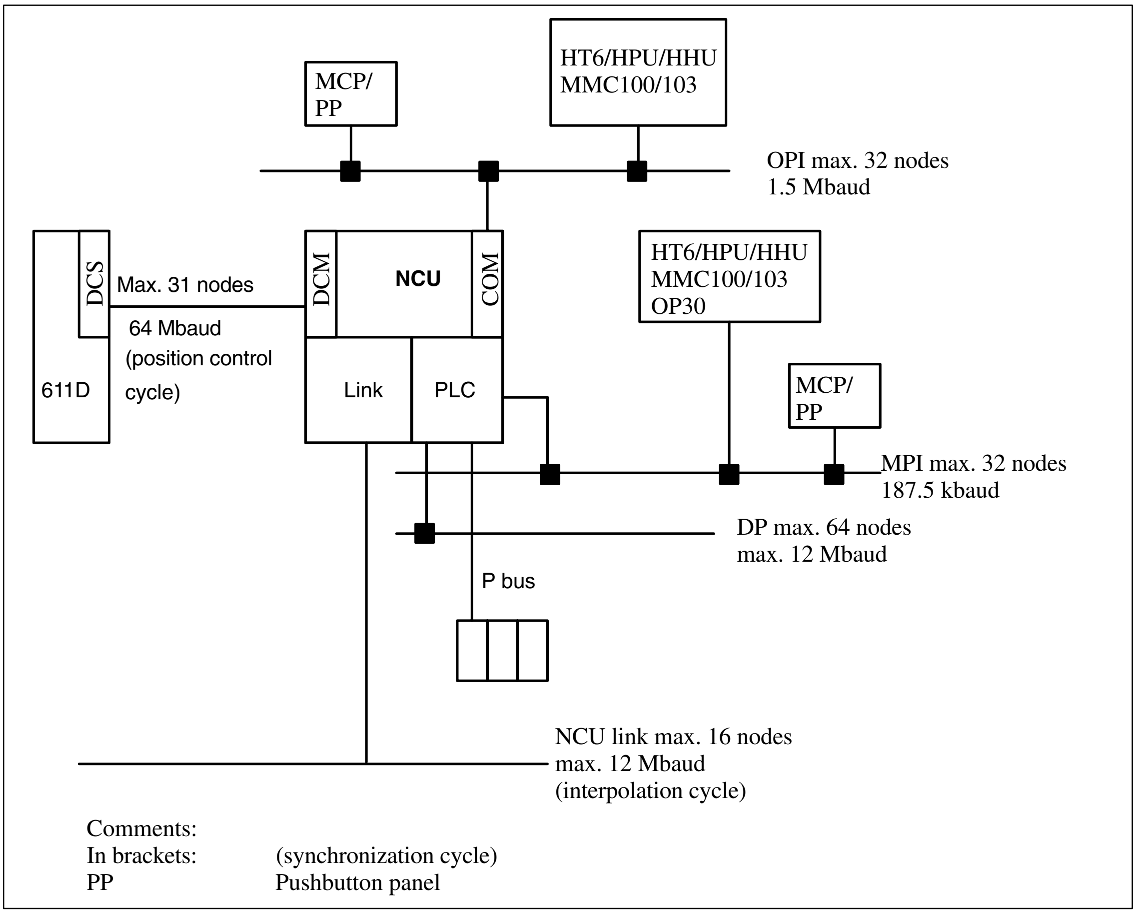



Fig. 1-2 Bus properties

**7-layer model structure**

Communication takes place on the following protocol layers:

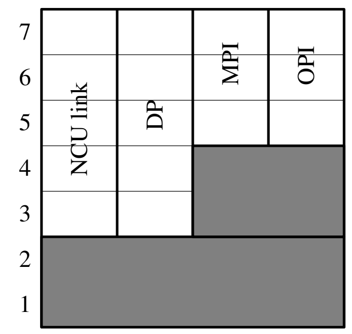



Fig. 1-3 Protocol levels in 7-layer model

The NCU link and DP can operate faster because they are assigned directly to layer 2.

## 1.2 Several operator panels and NCUs with control unit management (option)

**Introduction**

The plant configurations must be highly flexible to meet the requirements of complex machines such as rotary indexing machines, multi-spindle machines and complex NC production centers.
Often they need

- several (M) control units (MMC and MCP or HT6) owing to large dimensions of machine and physical separation of operator stations,
- several (N) NCUs owing to large number of axes and channels.

**Restriction**

The following subsection describes the additional functionality of software version 5. These functions are available only in connection with the **Control Unit Management Option**.

The standard functionality applies to all SW versions without the option. However different performance grades depending on SW version must also be taken into account; these are described in Chapter 3.

The standard functionality is described in the following subsection.

While the standard functionality supports only certain restricted combinations of MMCs and NCUs, the control unit management option provides a flexible, universal solution for satisfying the requirements above.

### 1.2.1 System features

**M:N concept**

The M:N concept is represented diagrammatically in Fig. 1-1 “Topology”:

Several control units (MMCs/HT6s and/or MCPs) are connected to several NCUs via a bus system. As the number of components is variable, they are given the indexes m and n; hence the name M:N concept.

This concept allows the user to connect any control units to any NCUs in the system (within the limits imposed by the hardware) via the bus and to switch them over as and when required.

**NCU link**

NCU link is an additional direct connection between the NCUs, enabling fast communication (see Sections 1.4 and 2.4).

**New features**

The features of SW 5 including the control unit management option are as follows:

- Independent connection of MMC and MCP
- Two independent MMC-PLC interfaces on each NCU for two autonomous MMC connections
  - MMC and MCP can be switched over together, or the MMC on its own. The HT6 basically constitutes a control unit comprising an MMC and MCP which must always be switched over together.
  - MMC states:
    - online/active: Operation and monitoring
    - online/passive: A window is displayed with header and alarm line and a message indicating "passive" state
    - offline
- Different bus systems (MPI/OPI) between MMC/MCP/HT6 and NCUs (changes only take effect after power-up)
- MMC function as server / as Main secondary operator panel
- A combination of both fixed MMCs and switchable MMCs can be connected.
- Suppression mechanism (priority-controlled) if more than two MMCs are competing for an NCU connection
- Up to 32 bus nodes (MMCs, MCPs, HT6s and/or NCUs, see Fig. 1-2)
- PLC controls the switchover process (control unit switchover to toolbox, directory PSP_PROG\m_zu_n.zip)
- Configuration file NETNAMES.INI with new parameters

**Supplementary conditions**

At any one time

- A max. of two MMCs/HT6s can be online on one NCU.
- Only one of them can be in an active state.
- The same value must be entered for the MMC and MCP addresses on the HT6. Since only values of 15 or less can be specified as MCP addresses, the MMC address is limited correspondingly.

## 1.2.2 Hardware

**Operator panel**

The OP030, OP031 or OP032 operator panel fronts feature a slimline screen, softkeys, a keyboard, interfaces and a power supply. In addition, an MMC 100.2/102 or 103 computer module is attached to OP031/32.

**Machine control panel**

The machine control panel (MCP) incorporates a keyboard, rotary button pad and interfaces.

**HT6**

The Handheld Terminal HT6 comprises a slimline screen, softkeys, keyboard, override rotary switch, emergency stop and enabling buttons as well as interfaces and power supply. The functions of the MMC and MCP are both integrated in the HT6.

**Difference between OP030 and OP031/32/HT6**

OP030 and OP031/32 differ in their assignment options to an NCU:

- OP030
  can be permanently assigned to one NCU only.
  It can be used as the secondoperator panel front for this NCU.
  The addresses of the connected partners can be set for this purpose.
- The OP031/32/HT6
  can have an active assignment to another NCU via
  HT6/MMC 100.2/102/103 operation.

**References**

The operator interfaces are described in the Operator's Guides of the operator panel fronts used.

/BA/, Operator's Guide
/FBO/, OP030 Operator's Guide
/BH/, Operator Components Manual

**Buses**

The control units (MMCs and/or MCPs, HT6s) and NCUs are connected by means of the

- MPI bus (MultiPoint Interface, 187.5 kbaud) or
- OPI bus (operator panel front interface, 1.5 Mbaud).

It is possible to combine different bus systems within one installation.

**Address assignments**

Bus nodes each have a unique address on the bus. The NCU uses

- a common address for the NC and PLC on the **OPI**,
- two separate addresses (for NC and PLC) on the **MPI interface**. The following applies:
  - SW 3.1 and later: PLC address can be reconfigured with STEP7. "2" is the default address for the PLC on the MPI.
  - SW 3.2 and later: As regards the addresses on the MPI interface, the following applies when the PLC-CPU 315 is used: NC address = PLC address + 1
  - SW 3.5 and later: As regards the addresses on the MPI interface, the following applies when the PLC-CPU 314 is used: NC address = PLC address + 1

**Defaults for OPI**

Addresses 0 (MMC) and 14 (HT6) are reserved at the works and 13 (NCK) is the default and these addresses should not be assigned to bus nodes in M:N systems.

- **Address 0** is reserved for PG diagnostics,

---

**Note**

The M:N switchover can malfunction when a PG is online. Remedy:
Either set the PG to Offline before switching the unit or connect it to the MPI.

---

- **Address 13** is the default for servicing/commissioning.
  This address can be reconfigured viaan MMC input. Reserved the address for NCU replacement if possible.

**Defaults for MPI**

- **Address 2** for PLC
- **Address 3** for NCU

**Number of active MMCs/HT6s on 1 NCU**

A maximum of two MMCs/HT6s (incl. COROS OPs) can be continually connected actively to one NCU. MMCs/HT6s on the OPI or MPI count in the same way.

**Number of MCPs/HHUs on 1 NCU**

As standard, two MCPs and one HHU can be connected to the OPI or MPI interface. The handheld terminal ((HT6) and the Handheld programming unit (HPU) count as one MMC and one MCP.

---

**Note**

The MPI/OPI network rules outlined in the “SINUMERIK 840D Installation Guide” Section 3.1 must be applied.

In particular, an M:N installation must be connected up by means of cables fitted with terminating resistors (identifiable by switch with which these are switched in and out).

---

## 1.2.3 Functions

**Defining properties**

The flexible M:N concept makes it possible to extensively modify the properties of the control units.

The assignment of the MMC properties can either be

- static or
- dynamic

**Static properties**

Static system properties are configured in file NETNAMES.INI (see paragraph below). They become effective at power-up and cannot be changed during the runtime:

- Assignment of bus nodes – bus system
- Combination of different bus systems (OPI, MPI)
- Assignment of MMCs – NCUs (which MMCs/HT6s can monitor which NCUs)
- MCP switchover
- Suppression priorities at changeover (see below)
- Utilization properties (see Chapter 2 "Properties of MMCs"):
  - Operator panel is the alarm/data management **server**
  - The control panel is mainor secondary **control panel**

**Dynamic properties**

The dynamic properties can be changed during runtime.

The states:

<table>
<tr><th colspan="2">Online</th><th>Offline</th></tr>
<tr><td colspan="2">Normal MMC operating mode with communication between the MMC/HT6 and NCU:<br>Operation and/or monitoring are possible.</td><td rowspan="3">No communication between the MMC/HT6 and NCU:<br>Operation and monitoring not possible.</td></tr>
<tr><th>Disable active</th><th>Passive</th></tr>
<tr><td>The operator can operate and monitor.</td><td>Operator cannot operate. He sees a window with header and alarm line and a message indicating "passive" state.</td></tr>
</table>

| | |
|---|---|
| Control unit switchover is **enabled**. | Control unit switchover is **disabled**. |

**Operating the M:N function**

The M:N function is operated via the "Channel menu" option.

The channel menu is selected via the "Switch over channel" key.
The horizontal softkeys are used to select a channel group (MMC 100.2/HT6: max. 8, MMC 102/103: max. 24 channel groups); up to eight connections to channels in different NCUs can be set up in one channel group.

The "Channel menu" screen displays all current connections and the associated symbol names.

---

**Note**

In the case of errors during power-up (for example, link setup fails), see Chapter 2 (Power-up).

---

**Suppression strategy**

If two MMCs/HT6s are online on one NCU, and a third MMC/HT6 would like to go online, then the latter can “suppress” one of the other two. Communication is then interrupted between this MMC and the NCU.

The algorithm responsible for this suppression is driven by priorities configured in the file NETNAMES.INI (see Subsection 2.1.8 “Suppression”).

### 1.2.4 Configurability

When the M:N system powers up, it must be aware of the existing control units, NCUs and communications links and their properties.

**NETNAMES.INI**

All this information is contained in the configuration file NETNAMES.INI which is configured before power-up.

This present description is mainly intended to provide the necessary knowledge for correctly setting up this configuration file for the M:N concept.

This means that

– the hardware configuration is displayed,

– the properties of the components are defined, and

– the desired switchovers/assignments are possible.

# 1.3 Several operator panel fronts and NCUs, standard functionality

The following applies to all M:N applications in which the control unit management option is not implemented. The level of performance is also dependent on SW version.

---

**Note**

Section 1.3 does not apply to the HT6, since only one HT6 can be operated on an NCU without control unit management.

---

## 1.3.1 System features

**SW 3.1 and higher**

Two MCPs and one HHU can be connected to the MPI or OPI.

The handheld programming unit (HPU) counts as an MMC and an MCP.

One of the panel fronts must be an OP030.

**SW 3.2 and higher**

The configurations "One operator panel front and up to three NCUs" are available.

The necessary configuration in the NC for the connection of MCPs/HHUs is made with the basic PLC routine (see Description of Functions, P3: Basic PLC program).

Address must be specified in the case of data exchange between PLCs via Profibus DP (PLC-CPU 315 only) or for global data (double addressing).

The following applies to PLC-CPU 315: NC address = PLC address + 1.

**SW 3.5 and higher**

The configurations "One operator panel front and up to four NCUs" and one MMC locally in each case are available.

## 1.3.2 Hardware

**Buses**

The connection between MMC 100 or MMC 102/103 (or other operator panel front CPUs) and NCUs is made via

- MPI bus (Multi- Point Interface, 187.5 kbaud) and/or
- OPI operator panel front interface, 1.5 Mbaud).

The buses can be used to link **m M**MCs and **n N**CU/PLC units.

**SW 3.1 and lower**

The machine control panel (MCP) must always be connected to connector X101 (OPI) on the NCU.

**SW 3.2 and higher**

Possibility of installing “Several operator panel fronts and NCUs” on the OPI (X101 on NCU) and the MPI (X122 on NCU).

When setting up the link via MPI, the PLC-CPU315 must be used if there are several NCUs, as it is possible to set the NC address with this PLC.

**SW 4 and lower**

Number of bus nodes: max. 16

**SW 4.1 and higher**

Number of bus nodes: max. 32

## 1.3.3 Functions

**SW 3.1 and higher**

“Several operator panel fronts and several NCUs” available in the basic version.

**SW 3.2 and higher**

Configuration in the NC for the connection of MCPs/HHUs is made with the basic PLC routine
(see Description of Functions, P3: Basic PLC program).

- Switchover of link to another NCU with the soft key labeled “conn...”:

A menu is overlaid where you can select the connections conn_1, ... conn_n (declared in NETNAMES.INI) via soft keys.
The name (name=...) allocated to the connection in NETNAMES.INI is displayed on the soft keys.

A connection to the new NCU is established by pressing the soft key.

- Changeover behavior on OP030:

It is not possible to change to another bus node online. The connection contained in NETNAMES.INI is permanently configured.

- Changeover behavior on MMC 100:

The “Conn...” soft key is only displayed if more than one link is configured in NETNAMES.INI.
When changing to the new NCU, the existing connection to another NCU is interrupted.

MMC applications, at the instant of link changeover, must no longer need the link to the previous NCU (e.g. for active data backup via RS-232 interface). Otherwise the control will issue a message if the connection is required.

With regard to the NCU to which the changeover takes place, the MMC behaves as if the system had been restarted. It is in the operating area selected as the Start operating area.

- Changeover behavior on MMC 102/103:

The “Connections” soft key is displayed only if the m to n function is activated on the control.
“m to n” is activated in menu “Start-up/MMC/Operator panel front”. Connections remain established with any changeover and the applications which have used these connections remain active. After the changeover, the MMC is in the same operating area with respect to the new NCU as it was previously with respect to another NCU.

- Possible defects

The NCU with which the connection is to be established can refuse the connection setup. Reason: NCU faulty or the NCU cannot operate with an additional MMC at this point.

The number of MMCs that an NCU can set up a connection with at any one time is configured in MD 10134 (MM_NUM_MMC_UNITS = number of MMC communications partners possible at any one time). OP030 requires one unit, MMC 100 or MMC 102/103 requires two units, as supplied. Other units (up to 12) are required for larger OEM packages.

- Alarms, messages

| MMC 100, OP030 | MMC 102/103 |
|---|---|
| It is only possible to output the alarms of the NCU with which a connection is currently established. | The alarms and messages of all connected NCUs can be processed simultaneously. |

- Alarm text management

| MMC 100, OP030 | MMC 102/103 |
|---|---|
| Only one version of the alarm texts can be stored on the operator component.<br>The standard alarm texts are stored once in the same formulation for all NCUs. The possible alarms for all connected NCUs must be stored in the one possible area for user alarms. | It is not possible to set up user alarm texts that apply specifically to the NCU<br>(MMC only manages one alarm text file, SW 3.4 MMC 102/103). |

- Connection check (MMC 100, MMC 102/103)

The address of a connected NCU (on OPI bus only) can be altered in the “Connections/Service” menu.
The new NCU address is stored on the NCU.

The soft key labeled “Service” is only displayed if the password for “Protection level Service” has been entered.

When the function for changing the address is started up, a direct connection between the MMC and the relevant NCU must always be established to ensure that the address is not programmed more than once on the bus.

**Note**

When the NCU is replaced (during servicing) or the back-up battery fails, the stored address is lost.

A general reset of the NCU does not delete the NCU address. The address can only be changed via the MMC.
The channel name should be assigned uniquely in MD 20000: CHAN_NAME (channel name).

**SW 3.2–3.x**

Operation of the M : N N

- MMC 102/103
  Select menu “Start-up/MMC/operator panel front” and choose between

  M: N (parameters for the MMC are stored in file NETNAMES.INI)
  and

  1 : 1 (addresses can be specified via the MMC).
- MMC 100
  The NETNAMES.INI file must be adapted by means of the application tool. The data are then transferred to the MMC.

**SW 3.5 and higher**

NC can exchange data with the PLC-CPU 314.

**SW 4 and higher**

The M:N function is operated via the “channel menu option”.

Precondition: Configuration of the NETNAMES.INI file (see /IAD/, Installation & Start-up Guide 840D, section on MMC).

The channel menu is selected via the “Switch over channel” key.
The horizontal softkeys are used to select a channel group (MMC 100/100.2: max. 8, MMC 102/103: max. 24 channel groups); up to eight connections to channels in different NCUs can be set up in one channel group.

The “Channel menu” screen displays all current connections and the associated symbol names.

**Note**

In the case of errors during power-up (for example, link setup fails), see Chapter 2 (Power-up).

### 1.3.4 Configurability

**From SW 3.1**
**2 MMCs : 1 NCU**

The option of creating a link between two operator panel fronts and one NCU has been implemented in SW 3.1, as the following diagram illustrates. In this case, there is a fixed assignment between the MCP and the NCU.

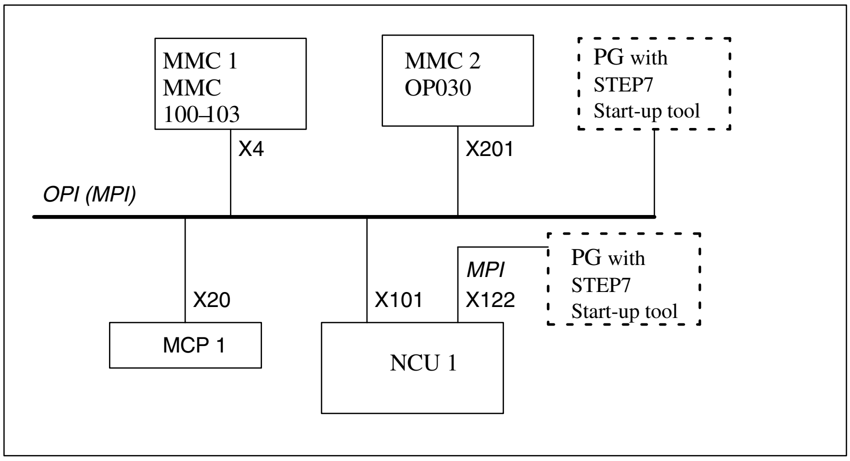



Fig. 1-4 Example configuration for SW 3.1 (m:n corresponds to 2:1)

The operator panel fronts, NCU and machine control panel are all either connected to the OPI bus or the MPI bus. A **homogenous** network must be provided with respect to these components. In SW 3.1, only OP030 may be used as the second operator panel front.

The configuration in the diagram would allow, for example, a high-performance operator panel front with MMC 100 or MMC 102/103 to be fitted on the front panel of a large machine tool and an OP030 operator panel front to be installed in the vicinity of secondary units (or on the rear panel of the machine

**Features**

When operating two operator panel fronts in the configuration illustrated above, the user will observe the following system operating characteristics:

1. For the NCU, there is no difference whether the input is from the MMC or OP030 operator panel fronts.
2. Each control unit can visualize selected displays independently of the other unit.
3. Spontaneous events such as alarms are displayed on both control units.
4. The protection level set on one MMC will also apply to the second MMC.
5. The system does not provide for any further co-ordination between the operator panels.

If the user applies the standard configuration shown in the diagram, then no further special settings are required.

**From SW 3.2**
**1 MMC : 3 NCUs**

In SW 3.2 and higher, it is possible to link one operator panel front and up to three NCUs (see Fig. 1-5). In this case, the MCP has a fixed assignment to the relevant NCU.

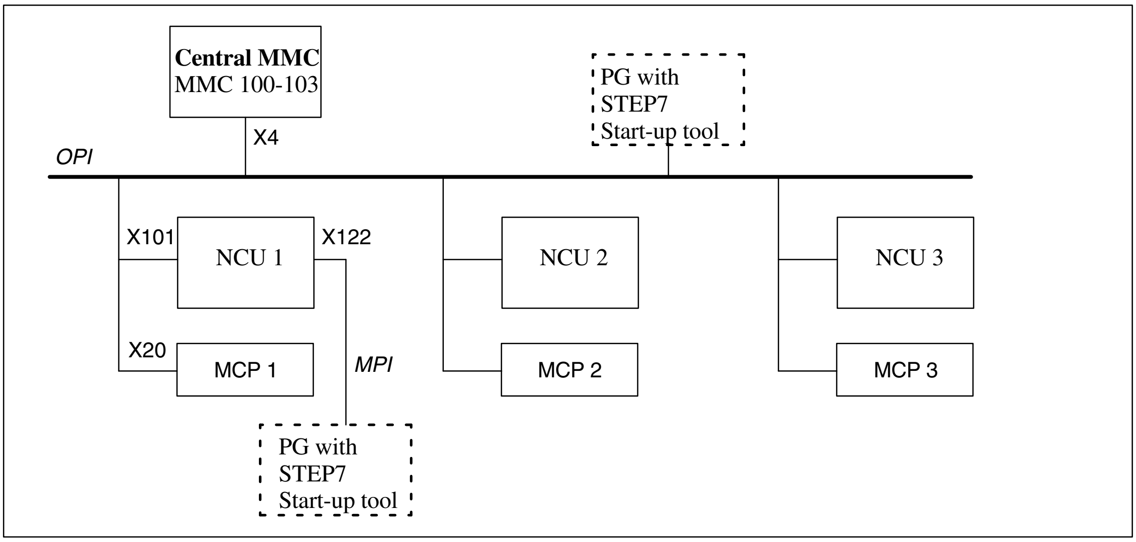



Fig. 1-5 Example configuration for SW 3.2 (m:n corresponds to 1:3)

It is possible to operate several NCUs from one MMC (several autonomous machines or one large machine with several NCUs). At any given time, only one preselected NCU is connected with the MMC for operating sequences. The MMC 100 is provided with one connection only for alarmswhereas the MMC 102/103 is connected to all NCUs for alarms.

**Features**

The operating characteristics  when several NCUs are linked to one MMC are as follows:

1. NCU operation:
   The user must select the NCU to be operated by means of a soft key.
   The operator display in the “Connection” operating area displays the name of the connection and the NCU to which the MMC is currently linked.
2. MMC 100:
   – No application should be active on the connection which is interrupted by the changeover to another NCU (Example: data backup via RS-232). System message “RS-232 active” is output if an attempt is made to change over the link when an application is active.
   – For the newly established connection, the MMC is positioned in the preset Start operating area (as with MMC restart).
3. MMC 102/103:
   When a link has been set up with another NCU, the last selected operating area (on the previous NCU) is immediately available for the new NCU.
4. PLC–CPU:
   – SW 3.2 and higher
     If the MPI bus is used (connector X122), the PLC must be a PLC-CPU315 to allow variable addressing for the PLC/NC.

– With SW 3.5 and higher
  the PLC-CPU314 can also be used.

**OEM solution**

As an OEM solution, an MMC 102/103 can be connected via an OPI to up to three NCUs (excluding 810D, as it does not have an OPI) as a program and alarm server (m=1, n=3).
In addition, a PG can be connected with a start-up tool.

---

**Note**

With SINUMERIK 810D, limited resources make it impossible to implement local MMCs.

---

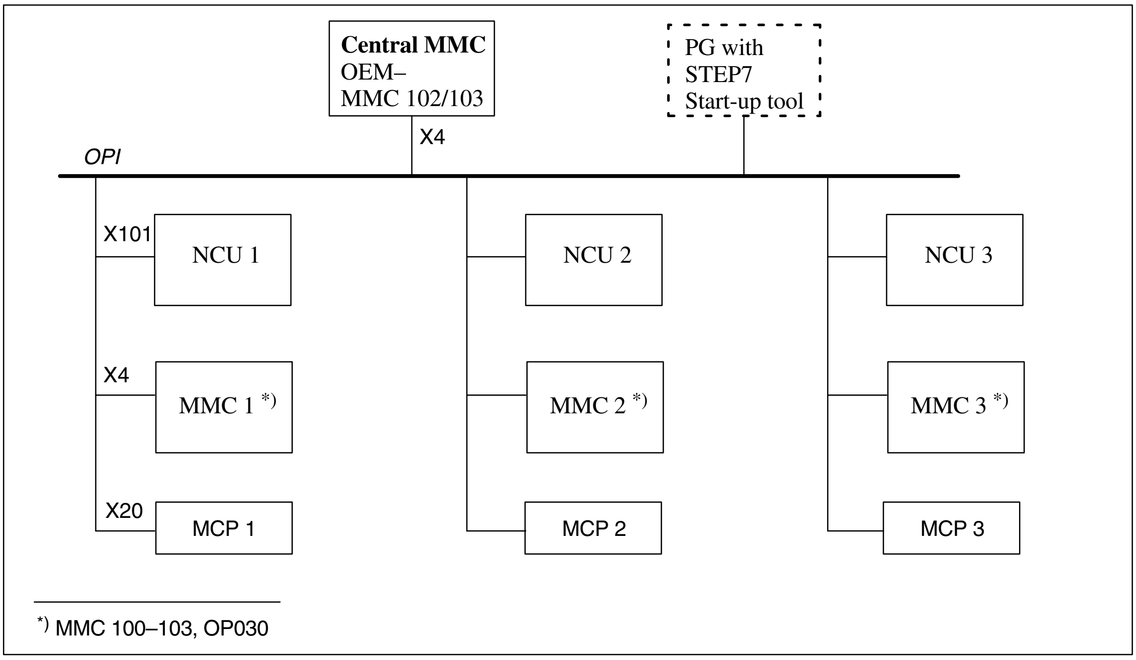



Fig. 1-6 OEM solution as example configuration for SW 3.2: Alarm and program server

**Features**

The following characteristics are typical of the OEM solution sketched in the above figure:

1. NCU operation:
   The user must select the NCU to be operated by means of a soft key. The operator screen displays the name of the connection and NCU with which the MMC is currently linked.
2. MMC 100
   Can only be connected to a local NCU.
3. MMC 102/103:
   When a link has been set up with another NCU, the last selected operating area (on the previous NCU) is immediately available for the new NCU.

**From SW 3.5**
**1 MMC : 4 NCUs**

In addition to the options described above for SW 3.5 and higher, it is also possible to create a link between an operator panel front (central MMC) and up to four NCUs, as illustrated in the following diagram. The MCP and the local MMC are permanently assigned to the relevant NCU.
A second MMC can be connected to the OPI.

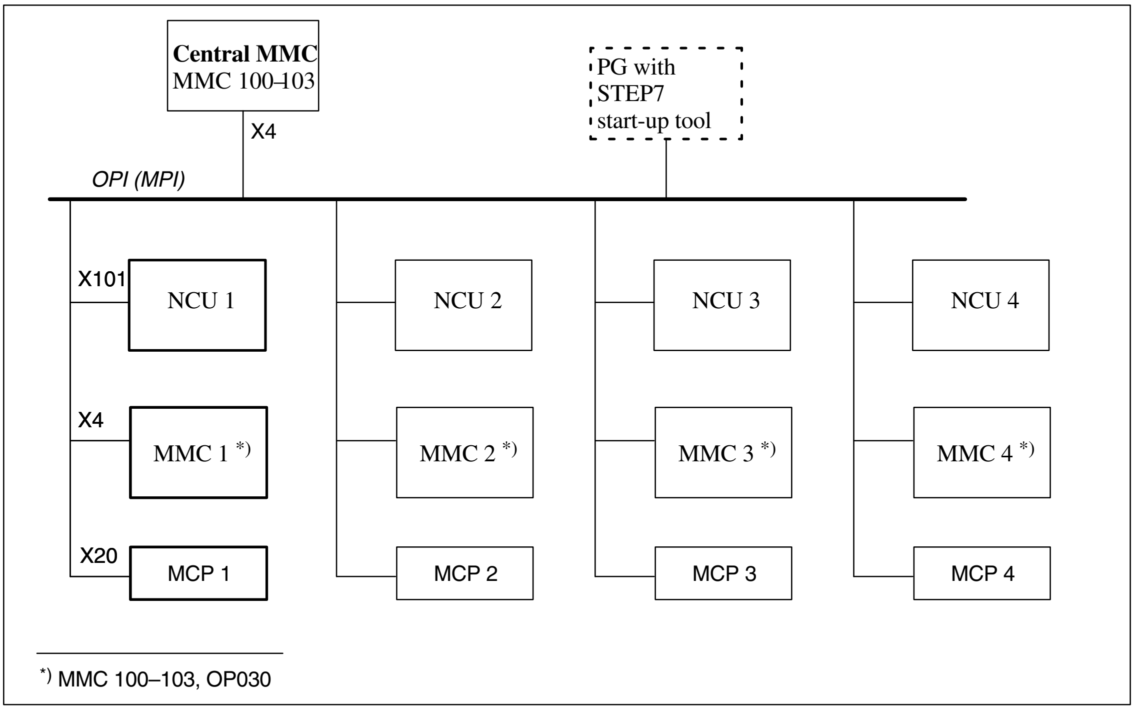



Fig. 1-7 Example configuration for SW 3.5 (m:n corresponds to 1:4)

It is possible to operate several NCUs from one MMC (several autonomous machines or one large machine with several NCUs). At any given time, only one preselected NCU is connected with the MMC for operating sequences. The MMC 100 is provided with one connection only for alarmswhereas the MMC 102/103 is connected to all NCUs for alarms.

---

**Note**

With SINUMERIK 810D, limited resources make it impossible to implement local MMCs.

---

**Required documentation**

**References:** /BH/, Operator Component Manual
/IAD/, Installation and Start-Up Guide
/FB/ P3, Basic PLC Program

The following is described in these documents:
– MPI/OPI Bus design, bus addresses, /IAD
– Bus termination, /IAD/, /FB/S7/
– Using basic PLC program to connect MCPs, /FB/P3/
– DIP-FIX settings on MCP, /IAD/

### 1.3.5 MPI/OPI network rules

Please take the following basic rules into account when undertaking network installations:

1. The bus line must be terminated at **both ends**. To do so, switch in the terminating resistor in the MPI connector in the first and last nodes. Switch off all other terminating resistors.

---

**Note**

- Only two inserted terminating resistors are permitted.
- Bus terminating resistors are **integral components** of the HHU/HPU devices.

---

2. At least 1 termination must be supplied with 5 V.
   This takes place automatically as soon as the MPI socket connector with its inserted terminating resistor is connected to a booted unit.
3. Drop cables (feeder cable from bus segment to node) should be as short as possible.

---

**Note**

Any spur lines that are not assigned should be removed if possible.

---

4. Every MPI node must **first** be connected and then activated.
   When disconnecting an MPI node, **first** deactivate the connection and then remove the connector.
5. One HHU and one HPU or two HHUs or two HPUs can be connected to each bus segment. Bus terminations must **not** be inserted in the distributor boxes of an HHU or HPU.
   If necessary, the connection can run from more than one HHU or HPU to a bus segment with intermediate repeater.
6. The following cable lengths for MPI or OPI for standard use without repeater may not be exceeded:

   MPI (187.5 kbaud): Max. total cable length is 1000 m

   OPI (1.5 Mbaud): Max. cable length in total 200 m

---

**Note**

Piggy-back connectors are not recommended for power connections.

---

You will find more information on bus communications in
**References**: /IAD/ Installation and Start-Up Guide 840D, Chapter 3 or
/IAC/ Installation and Start-Up Guide 810D, Chapter 3

## 1.4 NCU link

The NCU link, the link between several NCU units of an installation, is used in distributed system configurations.

**Introduction NCU link**

With high axis/channel requirements, for example, with rotary indexing machines and multi-spindle machines, the computing capacity can exceed the configuration possibilities and storage area offered by one single NCU. Several NCUs interconnected with an NCU link module provide a scalable solution which fully meets the requirements of this type of machine tools. The NCU link module offers fast NCU-NCU communication based on a synchronized 12MB PROFIBUS interface.

---

**Note**

NCU link is available in conjunction with MMC 103.

---

### 1.4.1 Types of distributed machines

**Machine characteristics**

Rotary indexing machines/multi-spindle machines have the following characteristics:

- Global, cross-station units (not assigned to one station):
  - Drum/rotary switching axis and
  - Units that go from station (position) to station (position) such as
    on rotary indexing machines:
    the rotary axis of theworkpiece clamping for
    multi-face machining operations
    on multi-spindle machines:
    spindle, quill
- Station-related (position-related), fixed-location units:
  - Slides, milling/drilling units used on the part that is changed from station to station for a machining task.

**Applications**

Rotary indexing machines (**RVM**) and multi-spindle machines (**MS**) are used as highly productive machines in the medium and large batch production. Their main advantage is that many machining steps can be performed on the workpiece in one clamp.

**NCU assignments**

According to the configuration of the RVM/MS, the many axes of these machines are assigned to different NCUs. (For example, one NCU for each machining unit or group of machining units). The global units are assigned to a separate NCU or distributed accordingly.

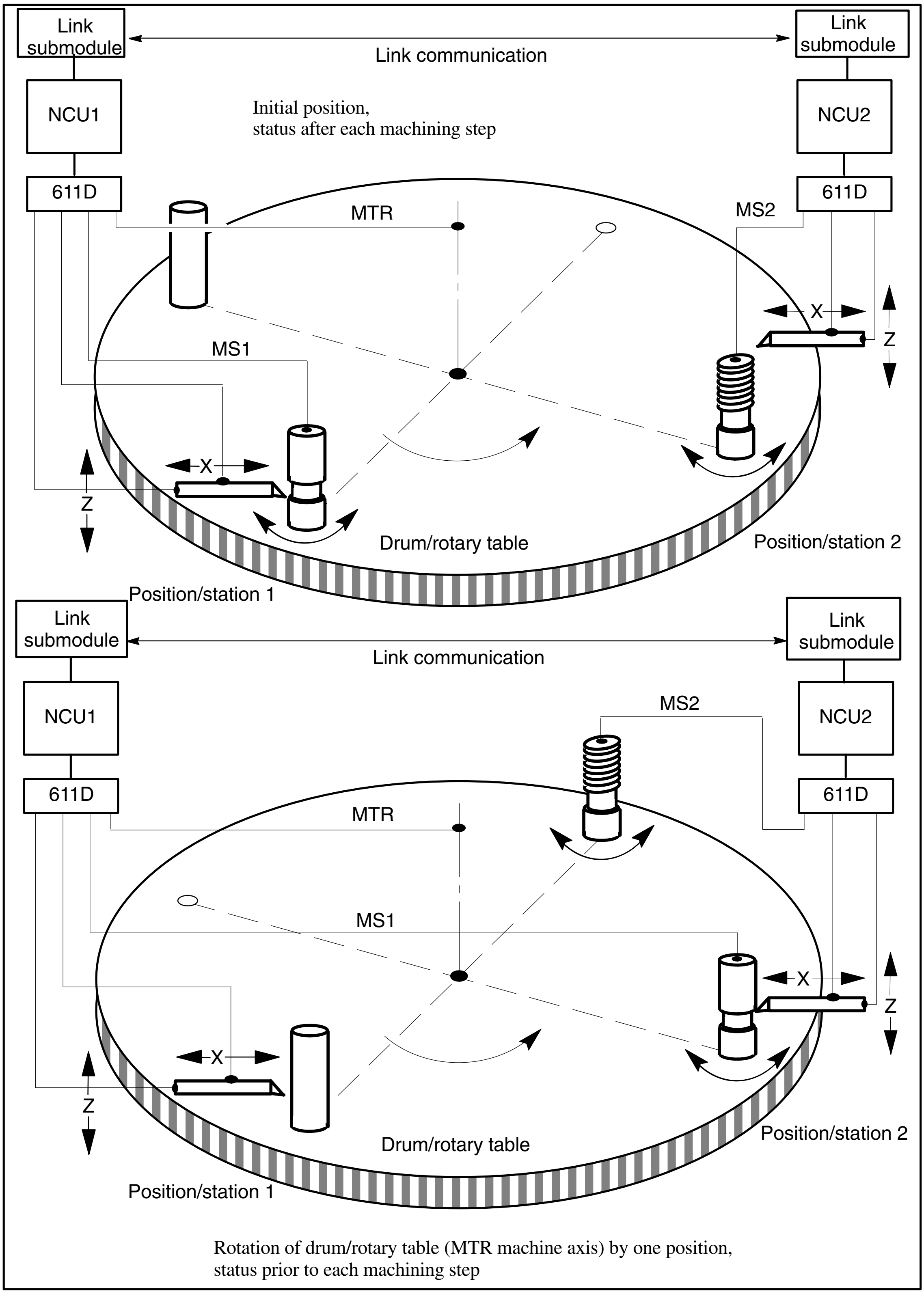



Fig. 1-8 Sectional diagram of a drum changeover

When advancing the rotary table with RVM or the drum with MS, the axis holding the workpiece moves to the next machining unit.

The axis holding the workpiece is now assigned to the machining unit's channel. This is on another channel in the example, but need not be.

As the above diagram "Drum changeover" shows, machine axis MS1, which is physically controlled by NCU1, is brought into position/station2 through rotation of the drum/rotary table. To ensure that a coordinated machining operation between the slide and spindle can now take place in position/station2, the commands for spindle MS1 are transferred in this position by means of link communication. Spindles MS1, MS2, ... are **link axes**.

Physical axes can only be subordinate to the motion control of one NCU channel at any one time. However, the motion control initiative for an axis can be assigned to different NCU channels in succession.

**Solutions**

To make a physical axis available to several different NCUs, the **Link Axis** property has been introduced. See Subsection 1.4.2 and Section 2.5. In the diagram above, MS1 becomes the link axis from the point of view of NCU2 (bottom diagram) after it has been turned to position/station2.

For variable assignment of channel axes to machine axes according to axis groups, the configuration concept **axis container** has been made available. See Subsection 1.4.3 and Section 2.6.

All link axes which are moved to the next position/station by a particular drum/rotary table must be managed in the same axis container.

## 1.4.2 Link axes

- Link axes
  - Definition:
    A link axis is an axis
    – whose drive control and position control are subordinate to another NCU or
    – which is the local axis of the NCU concerned, but can be addressed by another NCU.
  - The software option link axis must be installed.
- Coordination
  - The alternate use of a physical axis by several NCUs is dependent on all the relevant NCUs being aware of the status and data of the particular axis and on their ability to coordinate use of the axis.
- Interpolation
  - Local axes and link axes can be interpolated together through motion control by means of one NCU.

- Hardware
  - The NCUs involved in alternate use of axes across NCU limits must be equipped with a **link module**. The NCU link module offers fast NCU-to-NCU communication based on a synchronized 12-Mbaud Profibus interface.

**References:** /PHD/, Configuring Guide NCU 571–573.2

The following description provides the information required to configure, program and coordinate the distributed machines shown in the drawing.

For details please see Section 2.5.

### 1.4.3 Flexible configuration

**Introduction**

On rotary indexing machines/multi-spindle machines, the work-holding axes move from one machining unit to the next. As the machining units are under different NCU channels, it is necessary for the axes holding the workpiece to be dynamically reassigned to the appropriate NCU channel in the event of a station/position change. **Axis containers** are provided for this purpose.

Only one workpiece clamping axis/spindle is active at a time on the local machining unit. The axis container provides the possible connections to all clamping axes/spindles, of which exactly **one** is **activated** for the machining unit.

The following can be assigned via the axis container:

- local axes and/or
- link axes.

Switching between the available axes defined in an axis container works by cyclical shifting (rotation) of the entries in the axis container.

The modification can be triggered by the **parts program**.

**Scope of validity**

Axis containers with link axes are a NCU-cross device (NCU-global) that is coordinated via the control.
It is also possible to have axis containers that are only used for managing local axes.

For details about axis containers, please refer to Section 2.6.

## 1.4.4 User communication across the NCUs

**Definition**

- Link variables
  - Every NCU connected by means of a link module can address uniformly accessible **global link variables** for all connected NCUs. Link variables can be programmed in the same was as **system variables**. In general, these variables are defined and documented by the machine manufacturer.
  - Applications for link variables:
    – Global machine states
    – Workpiece clamping open/closed
    – ...
  - Data volume comparatively small
  - Transmission rate very high because information relevant to main run is available for exchange.
  - These system variables can be accessed from the **parts program** and from **synchronized actions**. You can configure the size of the memory area for global system variables.
  - All connected NCUs require one interpolation cycle before they can consistently read a new value in a global system variable.

For more information about the global system variables, please refer to 2.7.

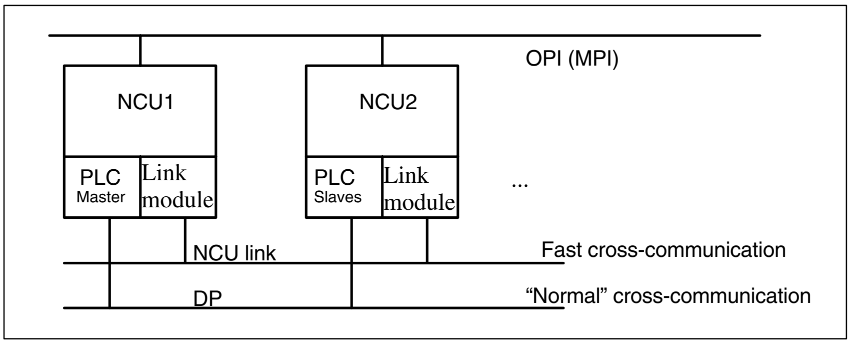



Fig. 1-9 Communication overview

---

**Note**

On installations without an NCU link, the link variables can also be used NCU-locally as an additional means of cross-channel communication.
In this instance, there is no interval of one interpolation cycle between writing and reading.

---

**References:** /FBSY/ Description of Functions Synchronized Actions

### 1.4.5 Lead link axes in software Version 6 and higher

The configuration illustrated below shows how to traverse following axes on several NCUs (NCU2 to NCU n in the diagram) in relation to the movement of the leading axis controlled by another NCU (NCU 1 in the example).

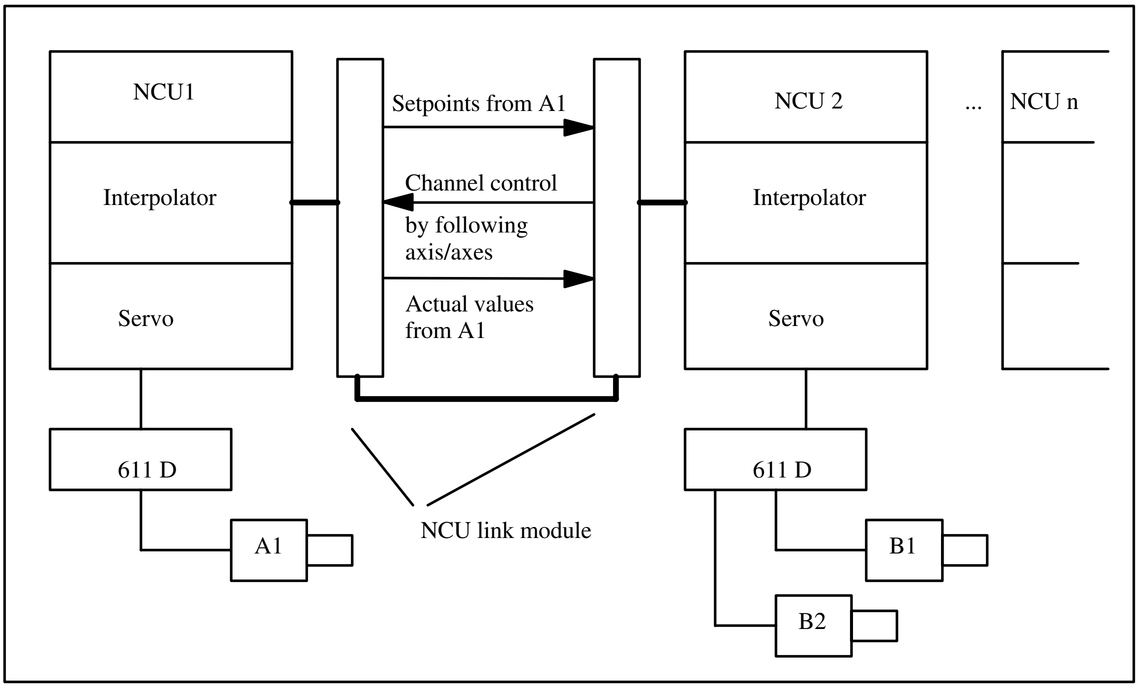



Fig. 1-10 NCU 2 moves the following axes via link to a leading axis on NCU1

If the boundary conditions described in Chapter 2 are complied with, it is possible for NCUs to traverse several leading axes in the whole NCU link group while the other NCUs derive the required following axis motions from them. The leading axes must only be moved by the NCUs to which they are physically connected (local NCUs). (Home NCUs).

### 1.4.6 NCU link with different interpolation cycles

**Introduction**

Connecting NCUs to link modules with different interpolation cycle settings offers additional application possibilities. This functionality is also called “Fast IPO link”, as when different cycles are set, one of the connected NCUs has the fastest interpolation cycle.

**Application guideline**

NCUs with a normal interpolation cycle drive axes and spindles with standard requirements of dynamic response and accuracy, while NCUs with the faster interpolation cycle can operate one or several axes which demand more in terms of dynamic response and accuracy.
Examples:
Eccentric turning (cams, pistons, or similar)
C and Z axes have normal requirements,
X axis with more exacting demands of dynamic response and accuracy

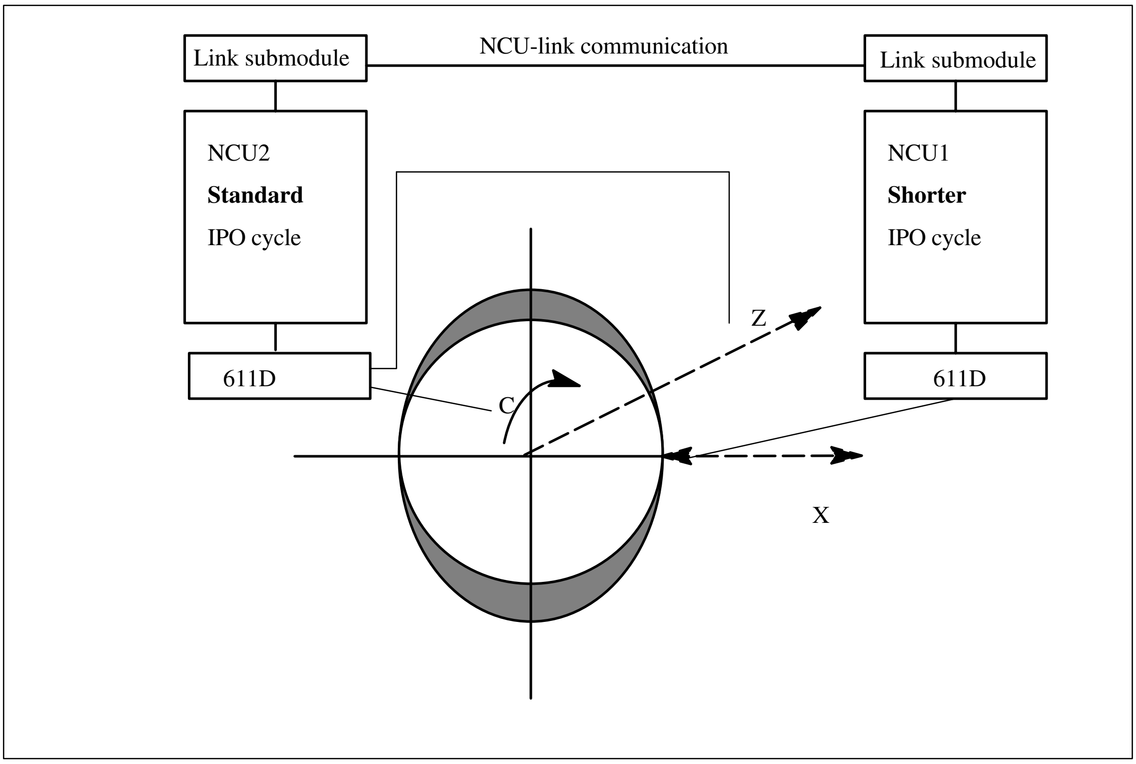



Fig. 1-11 NCU link with different interpolation cycles

### Main features

- Cross-NCU interpolation of fast (X) and standard (C,Z) axes/spindles (see Fig. 1-11).
- The parts program is running on the NCU with the faster interpolation cycle and can “see” the other axes as link axes or container link axes.
- Communication between the NCUs takes place via the link modules in the cycle of the slower NCU/NCUs (link cycle).
- The slower interpolation cycle is an integral multiple of the faster interpolation cycle.
- Owing to the different cycle clocks used, it is necessary to comply with the boundary conditions described in Section 2.11.
- Do not accelerate the slower interpolating axes during the machining operation to ensure that contour precision is provided between the faster and slower interpolating axes.

■

# Detailed Description 2

## 2.1 Several operator panel fronts and NCUs with control unit management option

The following chapter provides a detailed description of the preparations and implementation of the operating steps for the M:N concept.

Procedure

1. A number of different configurations are possible with the components of the existing system.
   The user selects one of these options to meet his individual requirements:
   - On the hardware side: by interconnecting components via bus systems
   - On the software side: by configuring static properties using configuration file NETNAMES.INI (see following paragraphs).
     These static properties are made operative during power-up and cannot be altered once the system is running.
   - Control unit switchover function in the PLCs of the relevant NCUs. The PLC control unit switchover function comprises several blocks. These perform the following tasks:
     – Check of switchover conditions
     – Prioritized suppression
     – Switchover

   The PLC SW “Control unit switchover” is supplied as part of the toolbox and can be parameterized if necessary. See Subsection 2.1.14.

   The option can be used only in PLCs with basic program version 05.03.01 or later.
2. Dynamic properties (such as online/offline states) can be changed when the system is running within the limits specified by the NETNAMES.INI file.

### 2.1.1 Hardware structure

As described in Chapter 1, a complex system can consist of M (several) control units and n (several) NCUs.
The diagram below illustrates a typical complex system:

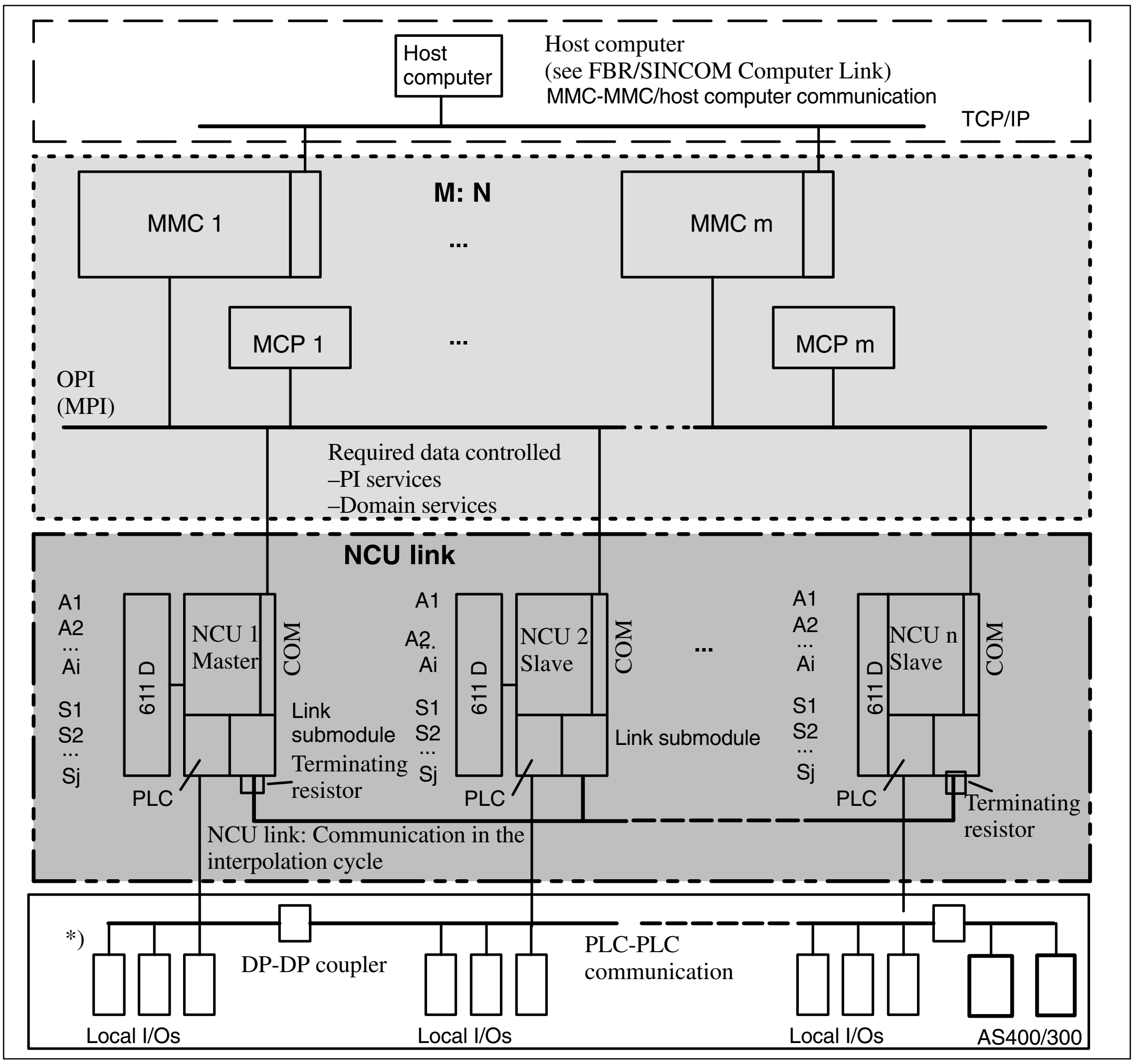



Fig. 2-1 Topology of distributed system configurations

*) PLC–PLC communication refers either to
– PLC–PLC cross-communication (master/slave comm.) or
– PLC local I/Os.

The hardware components are connected to one another via the bus (MPI and/or OPI). The relationships between the bus nodes (identification, properties, assignment and switchover) are software-controlled.

## 2.1.2 Features

**Client identification**

The assignment between bus nodes and the bus system is static and cannot be changed once the system is running. It is configured once in file NETNAMES.INI.

The client identification (CLIENT_IDENT) is composed of bus type and MMC bus address; the MMC uses it when logging on to an NCU to establish an online connection.

**Features**

The MMCs in an M:N installation have the following properties:

| **Server** | | **Control panel** | |
|---|---|---|---|
| Maintains a constant 1:N connection | | Can be switched to the different NCUs and maintains a constant 1:1 connection (only one at any one time!). Operator can operate and monitor. Connection is set up when the MMC goes online, and is disconnected when it goes offline. | |
| **Alarm server (MMC 103)** | **Data management server (MMC 103)** | **Main operator panel** | **Secondary operator panel** |
| Receives the alarms from all NCUs in an M:N installation. From its side, a constant 1:N connection is maintained. The process “Receive alarms” is always active and runs in the background. | Establishes all connections configured for it in NETNAMES.INI during booting and maintains a constant 1:N connection. Can receive, manage and distribute data within the framework of the job list concept. | Example: Main control panel for rotary index machines can be connected to all machining stations. | Example: Secondary control panel for rotary index machines can only be connected to one of two adjacent machining stations. |
| Cannot be suppressed (see Subsection 2.1.8) | Cannot be suppressed. | Cannot be suppressed. | Can be suppressed by the main or secondary control panel. |

Distribution of properties among the MMC types:

| Scope | MMC 102/103 | MMC 100.2/HT6 |
|---|---|---|
| Server | x | |
| Main control panel | x | x |
| Secondary control panel | x | x |

**MMC is both server and main control panel at the same time**

As a server, the MMC maintains constant 1:N connections; as a main control panel it has a switchable 1:1 connection.

If, as a control panel, the MMC is switched to another NCU, it occupies the same connection which it already has as a server. A new connection is not established.

**Permissible MMC combinations in an installation**

If there is a server (alarm/data management server) in an M:N installation, it is a main control panel at the same time.

In an M:N installation, there can only be one MMC/HT6 with the following properties:
Windows MMC (MMC 103) : Server and main control panel
or
Non-Windows MMC (MMC 100.2/HT6) : Main control panel

There can be any number of secondary control panels.

---

**Note**

For the function **Execution from external source** to be available, **one** operator panel in the system must be designated as a **server**. See below.

---

### 2.1.3 Configuration file NETNAMES.INI

**Configuration parameters**

As the hardware components can be freely combined (see the previous paragraph “Hardware structure”), it is necessary to provide the system with information about which components are connected, how they are connected to each other and how they interact.

In particular, it is necessary to regulate the competition among the different MMCs for the limited number of available interfaces (suppression, see 2.1.8).

Each MMC/HT6 has a configuration file NETNAMES.INI for this purpose; this is where the configuration parameters must be stored.

### 2.1.4 Structure of the configuration file

There is a separate configuration file NETNAMES.INI for **each** MMC/HT6.

It is structured as follows:

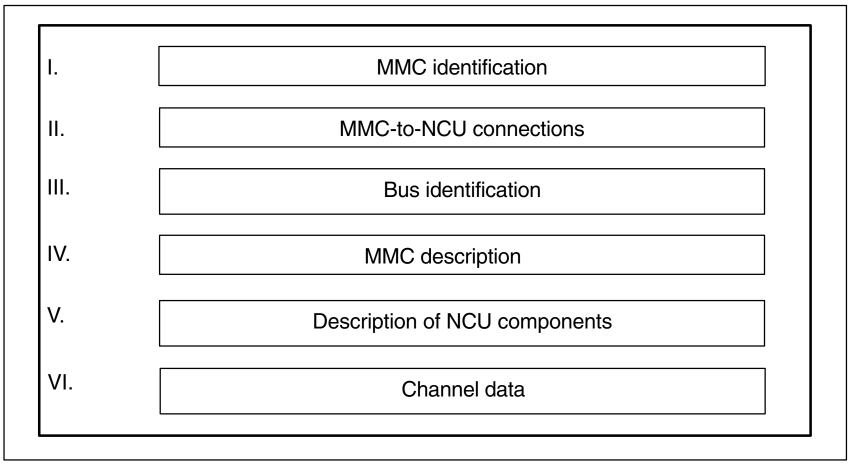



Fig. 2-2 Structure of the configuration file NETNAMES.INI

In the following tables,

– the components which the user may need to adapt or which can be freely named are shown in *italics*,

– alternative passwords are specified separated by |.

**I. MMC identification**

MMC/HT6 identifier to which NETNAMES.INI applies:

| Element | Explanation | Example |
|---|---|---|
| [own] | Header | [own] |
| owner = *Identifier* | MMC identification | owner = MMC_1 |

**II. MMC-to-NCU connections**

Configuring the connections between the MMC and the NCUs:

| Element | Explanation | Example |
|---|---|---|
| [conn *Identifier*] | Header | [conn MMC_1] |
| conn_i = *NCU_ID* | Configuring the NCU connection(s)<br>i = 1, ..., 15 | conn_1 = NCU_1<br>conn_2 = NCU_2<br>...<br>conn_i = NCU_i |

**III. Bus identification**

Defines which bus the MMC is attached to:

| Element | Explanation | Example |
|---|---|---|
| [param network] | Header | [param network] |
| bus = *OPI* \| *MPI* | Bus designation | bus = OPI |

opi: Operator panel front interface with 1.5 Mbaud
mpi: Multi-point interface with 187.5 kbaud

**Note**

The baud rate is automatically detected on the MMC 100.2/HT6.

**IV. MMC description**

Characterization of the MMC/HT6:

| Element | Explanation | Example |
|---|---|---|
| [param *Identifier*] | Header | [param MMC_1] |
| mmc_typ = *Type/connection identifier* | MMC characteristics (see below) | mmc_typ = 0x40<br>MMC is server and main operator panel<br>See below for explanations |
| mmc_bustyp = *OPI* \| *MPI* | Bus the MMC is attached to | mmc_bustyp = OPI |
| mmc_address = *address* | MMC_address | mmc_address = 2 |
| mstt_address or mcp_address = *address* | Address of MCPs to be switched simultaneously.<br>If not present, there is no MCP to be switched simul-taneously. | mstt_address = 6<br>or mcp_address = 6 |
| name = i*dentifier* | Any name allocated by the user (optional, max. 32 characters) | name = MMC_LINKS |
| start_mode = *ONLINE* \| *OFFLINE* | State after booting.<br>If ONLINE, link is set up via DEFAULT_channel entry to the associated NCU.<br>OFFLINE: No link is set up immediately after booting.<br>Important: In addition, it is necessary to set the entry NcddeDefaultMachineName = local in mmc.ini in the [GLOBAL] section. | start_mode = ONLINE (during power-up MMC is connected online to the NCU to which the channel is assigned via channel data (see VI )DE-FAULT_logChanGrp, DE-FAULT_ log_Chan). |

The integrated MCP is always switched over simultaneously with the HT6. The MCP address must have the same value as the MMC address. The MCP address must be set to values between 1 and 15.

**Note**

Note that the NCU configured via the DEFAULT channel must be the same as the NCU specified under NcddeDefaultMachineName in file **MMC.INI**.

Explanation for **mmc_typ**:
mmc_typ contains type and connection identifiers for the MMC and is transferred to the PLC at switching request. mmc_typ is evaluated as priority for the suppression strategy. See Subsection 2.1.8.

| | |
|---|---|
| Bit 7 = —— | (reserved) |
| Bit 6 = TRUE: | MMC is the server (MMC 103) and cannot be suppressed. |
| Bit 5 = TRUE: | MMC/HT6 is the main operator panel. |
| Bit 4 = TRUE: | MMC/HT6 is the secondary operator panel. |

The user can specify four additional MMC types which the control unit switchover function of the PLC takes into account in its suppression strategy:

| | |
|---|---|
| Bit 3 = TRUE: | OEM_MMC_3 |
| Bit 2 = TRUE: | OEM_MMC_2 |
| Bit 1 = TRUE: | OEM_MMC_1 |
| Bit 0 = TRUE: | OEM_MMC_0 |

If no mmc_typ is entered in file NETNAMES.INI, then the MMC/HT6 powers up by the method for standard functionality.

### V. Description of the NCU component(s)

A separate entry must be generated for every single NCU component connected to the bus.

| Element | Explanation | Example |
|---|---|---|
| [param *NCU_ID*] | Header | [param NCU_1] |
| name = *any_name* | Any name assigned by the user; is output in the alarm line (optional, max. 32 characters) | name = NCU1 |
| type= *NCU_570 / NCU_571 / NCU_572 / NCU_573* | NCU type | type = NCU_572 |
| nck_address = *j* | Address of NCU component on the bus: *j* = 1, 2, ..., 30 *) | nck_address = 14 |
| plc_address = *p* | Address of PLC component on the bus: *p* = 1, 2, ..., 30 *) (only necessary for the MPI bus because j = p) for the OPI bus) | plc_address = 14 |

*) With the MPI bus:
Since the associated NCU always occupies the next-higher address than the PLC, the PLC address must not be 31. Address 31 can, for example, be assigned to an MMC.

---

**Note**

If the bus node addresses on the MPI bus are configured in conformance with SIMATIC, the configuring engineer can read out the assigned addresses using a SIMATIC programming device and use them to create the NETNAMES.INI file.

---

### VI. Channel data

The control unit switchover option can work only if the control unit knows how channels are assigned to NCUs so that it can set up links between the control unit and NCUs. (Channel menu).

**Concept**

The following operations must be performed:

1. Definition of technological channel groups
2. Assignment of channels to groups
3. Assignment of NCUs to channels
4. Definition of power-up link

NCUs are addressed indirectly on the basis of channel group and channel on the control unit. See 2.1.11 Operator interface.

**References:** /IAM/, MMC Installation and Start-Up Guide
/BA/, Operator's Guide
/S7HR/, SIMATIC S7-300
/FB/, P3 "Basic PLC Program"

| Element | Explanation | Example |
|---|---|---|
| [chan *identifier*] | Header (channel menu of MMC_1) | [chan MMC_1] |
| DEFAULT_logChanGrp = *group* | Channel group of channel during power-up (4.) | DEFAULT_logChanGrp = Mill1 |
| DEFAULT_logChan = *channel* | Selected channel during power-up (4.) | DEFAULT_logChan = channel11 |
| ShowChanMenu = *TRUE / FALSE* | TRUE Display channel menu | ShowChanMenu = TRUE |
| logChanSetList = *group list* | List of channel groups (1.) | logChanSet = mill1, mill2 |
| [*group*] | Head (2.) | [mill1] |
| logChanList = *channel1, channel2,...* | Groups channels separated by comma (2.) | logChanList = channel11, channel12, channel13 |
| [*channel*] | Head (3.) | [channel11] |
| logNCName = *identifier* | Log. identifier of an NCU (3.) | logNCName = NCU_1 |
| ChanNum = i (i = 1, 2, 3,...) | Number of channel configured for associated NCU (3.) | ChanNum = 1 |
| And so on for all channels in group | | |
| Continue with next group and its channels | | |

A complete example of how to configure the channel menu can be found in 6.1.

## 2.1.5 Creating and using the configuration file

**Syntax**

The configuration file must be generated as an ASCII file. The syntax is the same as that used in Windows *.ini files.

In particular, the following is applicable:

- Passwords must be typed in small letters.
- Comments can be inserted in the parameter file (limited on the left by ";" and on the right by end of line).
- Blanks may be used as separators at any position except in identifiers and passwords.

**MMC 100.2, OP030, HT6**

The NETNAMES.INI file generated on the PC/PG is loaded, as described in

**References:** /IK/, Installation Kit

via the RS-232 interface and permanently stored in the FLASH memory of the control units.

**MMC 102/103**

The NETNAMES.INI file can be processed directly with an editor (in menu "Start-up/MMC/Editor" or DOS_SHELL) on the hard disk of the operator component. The NETNAMES.INI file is stored in the installation directory C:\USER\.

**Example**

For a sample configuration file, please refer to Chapter 6.

## 2.1.6 Booting

**Defaults standard functionality**

The following defaults are applied (standard M:N = 1:1) if no NETNAMES.INI configuring file is loaded into the MMC 100.2/OP030/HT6 or if the latter cannot be interpreted:

- – The bus type used is automatically determined.
- – MMC has address 1.
- – OP030 has address 10.
- – NCU and PLC both have address 13 for an OPI bus.
- – NCU has address 3 and PLC address 2 for an MPI bus.

**With option**

If, however, a special NETNAMES.INI file is created, then it must correspond exactly to the actual network on account of the special features described below.

If an M:N-capable MMC fails to set up a link to the NCU during power-up or in the case of a configuring error, the MMC switches over to OFFLINE operating mode. In this MMC mode the operator can switch over to the area application via the Recall key and then to the start-up area.

**Compatibility**

The use of the above defaults establishes compatibility with earlier software versions for operation of the panel front.

**Power-up with MMC 100.2/HT6**

An MMC 100.2/HT6 control unit can only set up an active link to the NCU if the configuration in NETNAMES.INI is correct as described in Section “Structure of the configuration file”. MMC 100, HT6 and OP030 can power up parallel on one NCU, because as bus nodes they have different addresses.
The OP030 can be used as a second operator panel front that has a fixed assignment to an NCU.

If the configured addresses do not match the real addresses (NC/PLC address), the start-up engineer can use the following key sequence to power-up an NCU that is not configured.

**Sequence**

1. MMC boots on the NCU with bus address 13, if the NETNAMES.INI was not changed (original works settings).
2. File NETNAMES.INI has been altered, message
   “MMC 100 version xx.xx.xx: waiting for connection ...”
   – Press key “1”, the message:
     “choice: ’1’=set new start-address, ’^’ =boot” is displayed.
   – Press the “1” key, the bus addresses of all nodes connected to the bus are displayed. The message:
     “Please try one of the shown addresses or press ’^’ to reboot
     ’1’,_,_,_,_,’6’,_,..._,’D’,_,...” is displayed.
   – Press “D” key and INPUT
   – MMC/HT6 boots on the NCU with bus address 13 (if an NCU is configured under the address found).
3. Enter new NC address in the Start-up/NC/NC address operating area and confirm with “Yes”.
4. NC reset (new address is only valid after NC reset)
5. Configure connection/channel menu in the NETNAMES.INI file and transfer to the MMC.
6. After the NCU addresses have been assigned, the bus can be wired for m:n operation.

---

**Note**

You can operate an OP030 and an MMC 100.2/HT6 on an interface without assigning parameters (various bus addresses are available in the delivery state).

---

**Power-up with MMC 102/103, standard solution**

The MMC 102/103 can power up even if the link to the NCU cannot be made due to errors in the configuring parameters.
An NCU address can be specified explicitly through the entry of a “1 : 1“ connection in the “Start-up/MMC operator panel front” menu. When the MMC has powered up again, the communications link between the MMC and NCU/PLC will work properly.

**Sequence**

1. MMC boots on the NCU with bus address 13, if the NETNAMES.INI was not changed (original works settings).
2. NCU bus address was changed, the following alarm is output
   "120201 name: communication failed"
   – Set the connection to 1:1 in the Start-up/MMC/Operator panel front operating area and enter "13" as the NC address
   – Confirm with OK and boot the MMC
3. – 6. As on MMC 100.2

---

**Note**

In the event of an error, check
– the active bus nodes in the menu
– start-up/NC/NCK addresses (MMC 100.2, HT6 and 103),
– start-up/MMC/operator panel front (MMC 103).

---

**Power-up with MMC 102/103, Option**

If the control unit switchover option is installed, a configuring problem can be corrected as follows:

1. Select the channel menu with the input key
2. Go to the area switchover screen by pressing Recall
3. Select start-up.

**Required documentation**

**References:** /BH/, Operator Components Manual
/IAD/, Installation & Start-up Guide
/FB/ P3, Basic PLC Program

In this document you will find the following described:
– Creation of MPI/OPI bus link, bus addresses, /IAD/
– Bus terminator, /IAD/, /FB/S7
– Using basic PLC program to connect MCPs, /FB/, P3
– DIP-FIX settings on the MCP, /IAD/

---

**Note**

After performing a series machine start-up, a Power On must be performed on the MMC/HMI (PCU50) so that the bus nodes (PLC, NC, MMC/HMI) can synchronize again.

---

## 2.1.7 MMC switchover

With the M:N concept, you can change the control unit properties and states configured in the NETNAMES.INI file during operation.

For example, the user can

- change MMCs (see Subsection 2.1.9),
- change MCPs (see Subsection 2.1.13).

Up to two MMCs can be online at the same time on one NCU. A suppression strategy (see Subsection 2.1.8) is provided to avoid conflicts when more than two MMCs want to go online on one NCU.

The MMC properties are configured for each MMC in the NETNAMES.INI file. If an MMC wants to go online on an NCU via the switchover protocol, its parameters are passed on to the PLC of the respective NCU. The PLC program **Control Unit Switchover** evaluates the parameters:

– Check suppression conditions

– Switchover if necessary

### 2.1.8 Forced break

Up to two MMCs or HT6s can be online on each NCU. If this is the case, and another MMC/HT6 would like to go online, it is necessary to ensure that there are no conflicts. This is achieved by means of the suppression algorithm described below.

**Sequence**

- The PLC sends an offline request to the MMC to be suppressed.
- It returns a positive or negative acknowledgement to the PLC:
  – If it is positive, the MMC/HT6 is suppressed (see below). It terminates the communication with the NCU and goes into offline mode.

    Any MCP assigned to the MMC is deactivated by the PLC.

    The integrated MCP is always assigned on the HT6 and is thus also deactivated.
  – A negative acknowledgement is output if processes run on the MMC that cannot be interrupted, e.g. operation via RS-232 or data transfer between the NCU and MMC.

    In this case the MMC/HT6 is not suppressed; it remains online on this NCU.

**Suppression strategy**

The PLC program “Control Unit Switchover” operates according to the

– priorities of the control units and

– the active processes

The priority depends on parameter **mmc_typ** in configuring file NETNAMES.INI (see paragraph above “Structure of the configuration file”).

If an MMC/HT6 wants to go online to an NCU, it stores mmc_typ (priority) in its MMC_PLC interface. The Control Unit Switchover program evaluates this according to the following table:

| MMC property | Priority |
|---|---|
| Server | 6 |
| Main control panel | 5 |
| Secondary control panel | 4 |
| OEM-MMC 3 | 3 |
| OEM-MMC 2 | 2 |
| OEM-MMC 1 | 1 |
| OEM-MMC 0 | 0 |

**Suppression rules**

The following rules apply for the MMC suppression:

- High priority suppresses lower or equal priority subject to the following supplementary conditions:
  - Servers cannot be suppressed, as they require a permanent connection to each NCU.
  - MMCs/HT6s on which the following processes are active cannot be suppressed:

    – Data transfer, e.g. from/to NCU

    – MMC/HT6 is in the process of switching to the relevant NCU

    – MMC/HT6 is just changing operating mode

    –OEM disables switchover
- Equal priority of nodes between active MMC/HT6 and competitor MMC:
  - The active MMC/HT6 is suppressed

## 2.1.9 Connection and switchover conditions

In order to

– allow an MMC/HT6 which is currently working offline to go online on a particular NCU or

– switch an MMC/HT6 which is working online over to another NCU,

1. Call the channel menu on this MMC by pressing the channel switchover key (applies only to MMC).
2. Select the channel group via a horizontal soft key.
3. Select the appropriate vertical soft key for the channel. See 2.1.10.

HT6: 1. Activate the “Panel Function” by selecting the key with the same name.
2. Select the “Channel” soft key.
3. Select the channel group.
4. Select the channel.
If the chosen channel is not included in this group, then you can return to point 2. by pressing the “Recall” key.

The MMC/HT6 is then switched to online operation or to another NCU, provided that its change in status is not blocked by one of the following conditions (displayed in message line).

Table 2-1 Messages associated with MMC switchover (MMC 103 without message number)

| MMC100 | Message text |
|---|---|
| 109001 | No switchover: Switchover disable set in current PLC |
| 109002 | No switchover: Target PLC occupied, try again |
| 109003 | No switchover: Switchover disable set in target PLC |
| 109004 | No switchover: PLC occupied by higher-priority MMCs |
| 109005 | No switchover: No MMC on target PLC can be suppressed |
| 109006 | No switchover: Select channel invalid |
| 109007 | Channel switchover in progress |
| 109009 | Switchover: Error in internal state |
| 109010 | Suppression: Error in internal state |
| 109012 | Control unit switchover, PLC timeout: 002 |
| 109013 | Activation rejected |

**Note**

Corresponding messages are output without a message number on the MMC 103.

Additional messages can be generated in the MMC 100.2/HT6 and MMC 103 indicating the current status or errors in the configuration or the operating sequence.

For details see
**References:** /DA/, Diagnostics Guide, Chapter 1

## 2.1.10 Implementation of control unit switchover

Control unit switchover is an extension of channel switchover.

**Channel switchover**

“Channel switchover” is a configuring means by which channels of any chosen NCUs can be grouped and named individually. MMC switchover to another NCU is implemented as part of channel switchover functionality.

Configuring of channels is based on file NETNAMES.INI. See 2.1.4.

## 2.1.11 User interface

**Function**

You can establish a connection between the MMC unit and one of the connected NCU/PLC units in every operating area.

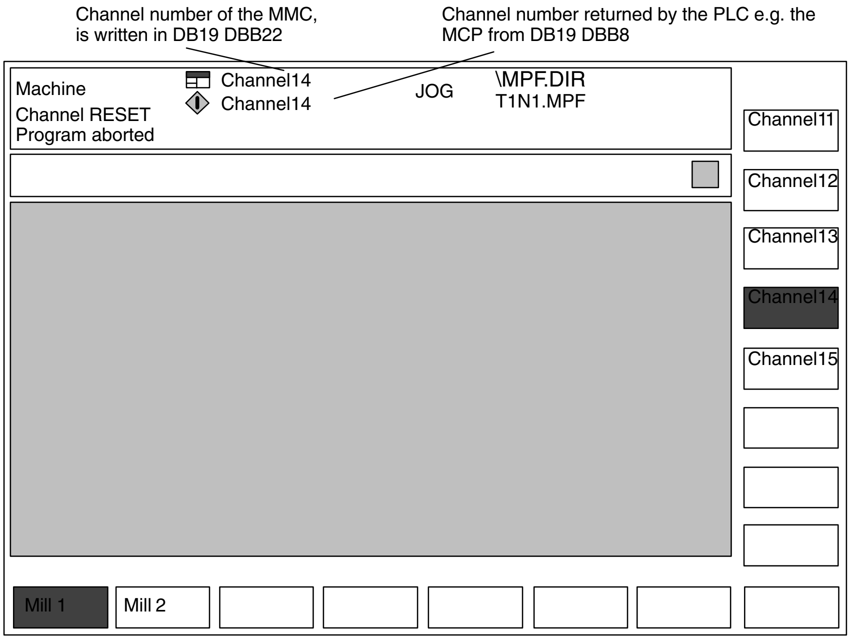



Fig. 2-3 Channel menu (the comments refer to the 1st MMC interface)

Only the channels of the respective group are displayed.

Activate the channel changeover key. The currently existing connection is displayed by means of the highlighted softkeys (horizontal, vertical) if the channel menu is active.

**Channel switchover**

You can switch to other channels by means of the vertically arranged softkeys.

**Group switchover**

You can switch to another group by means of the softkeys on the horizontal menu (see previous Section); the channels of the currently selected group are now displayed on the vertical soft keys. Switchover to another channel (and if necessary to another NC) only takes place upon activation of a vertical soft key.

**NC switchover**

You can change to another NC via the vertical softkeys if the channel is not on the current NC.
Procedure: Configure a channel area NCs (horiz. softkeys 1–8) if applicable and link a channel to vertical softkeys from every NCU.

---

**Note**

The soft keys only offer the connections that are really assigned and whose channels are active in the respective NC.

HT6

The channel menu on the HT6 is a two-level structure. In the first stage you select the channel group and in the second stage, the channel. For details please refer to the HT6 Operator's Guide.

---

### 2.1.12 Operating mode switchover

Two MMCs/HT6s can be online at the same time on one NCU. In order to avoid both gaining write access to the same data or file simultaneously, there are two operating modes, i.e.:

- the active and
- the passive operating mode.

Only one of the two MMCs/HT6s can be active; the other is passive.

The interaction takes place according to the following rules:

**Active operating mode**

- The user requests active operating mode by pressing a key on the operator panel front.
  Active mode has the following characteristics:
  - All operations and operating areas are activated.
  - The operator can operate and monitor.
  - The MCP assigned to the MMC/HT6 is activated.
  - If data transfer processes (e.g. series machine start-up, various tool management services, start-up of the drive configuration) are running between the other MMC/HT6 and the joint NCU, it cannot become active immediately.

**Passive operating mode**

- Passive mode is effective when the other MMC/HT6 has requested active mode.
  The features of this mode are as follows:
  - The connection to the NCU remains established.
  - All operations are deactivated.
  - Operator cannot operate. A window is displayed with header and alarm line and a message indicating "passive" state.
  - The global menu is activated.
  - Any services initiated before (in active mode) remain active (e.g. operation via RS-232, reloading parts programs, executing the job list, alarms).
  - The MCP assigned to the MMC/HT6 is deactivated.
  - The application window and softkeys are disabled.

The active operating mode can be selected by 2 different methods:

- Input key
- Channel switchover key and channel selection.

**Rules for operating mode changeover**

The following rules apply to operating mode changeovers
(see also Subsection 2.1.8 “Suppression strategy“):

- An MMC/HT6 which goes online to an NCU is assigned active operating mode on this NCU.
  If another MMC/HT6 was previously active on this same NCU, it switches to passive mode if permitted by the PLC.
- If two MMCs/HT6s are online, the operating mode is changed by pressing the key (“Input”, ENTER, RETURN) used to select the active operating mode.
- Changeover from the active to the passive operating mode might be rejected by the MMC/HT6 if the current MMC application cannot be aborted or is still in progress. Likewise, active mode cannot be selected on an MMC/HT6 if the other MMC/HT6 currently linked to the NCU cannot be switched to passive mode.
- If an online request is issued by an MMC/HT6
  – and no MMC/HT6 is yet online:
    The MMC/HT6 issuing the request goes online and switches to active mode.
    Any assigned MCP is activated by the PLC.
  – and an MMC/HT6 is already online:
    This MMC/HT6 switches to passive mode.
    The MMC/HT6 issuing the request goes online and is given active mode.
  – and two MMCs/HT6s are already online (both of secondary control panel type):
    The currently active MMC/HT6 switches to passive mode and is suppressed.
    The MMC/HT6 issuing the request goes online and is given active mode.
  – and two MMCs/HT6s are already online (one of main control panel type and in active mode, the other of secondary control panel type and in passive mode):
    The currently active MMC/HT6 switches to passive mode. The currently passive MMC/HT6 is suppressed.
    The MMC/HT6 issuing the request goes online and is given active mode.
  – and two MMCs/HT6s are already online (one of secondary control panel type and in active mode, the other of main control panel type and in passive mode):
    The currently active MMC/HT6 switches to passive mode and is suppressed.
    The MMC/HT6 issuing the request goes online and is given active mode.
- If two MMCs are online on one NCU and the active MMC/HT6 goes offline, it first switches to passive mode. Then the second MMC/HT6 switches to active mode and the first one disconnects the link to the NCU.

**Note**

The MMC type is assessed as priority for the suppression strategy. See 2.1.8.

If the active MMC/HT6 cannot be switched to passive mode, then the competing MMC/HT6 is switched to passive mode.

## 2.1.13 MCP switchover

An MCP cannot be switched over independently of the MMC/HT6 it is assigned to. It can be switched over only if

- the MMC switches over and
- the MCP address is stored in the MMC parameter block or the MMC-PLC interface (see paragraph “Structure of the configuration file” in this section).
- MCP_enable is set in the control unit switchover function on the PLC.

**Activating/ deactivating the MCP**

If an MCP is assigned to the MMC/HT6 in the NETNAMES.INI file, it is activated/deactivated as part of the operating mode change. The MCP switchover in the PLC is called by the operating mode change as a subfunction. The parameters for the MCP switchover are stored in the MMC-PLC interface.

| MMC/HT6 is changing operating mode | MCP is |
|---|---|
| active –> passive | deactivated |
| passive –> active | activated |

## 2.1.14 PLC program “Control Unit Switchover”

**Introduction**

“Control unit switchover” is an important controlling function in the overall M:N strategy:

– MMC/HT6 makes requests regarding the dynamic assignment of MMCs/HT6s to NCUs according to the configured options in NETNAMES.INI and displays information about existing links.

– The PLC control unit switchover checks the priorities of the requests and the states of the components involved and switches over if necessary.

– The NCU sets signals and evaluates signals required in connection with the control unit switchover function.

The control unit switchover function is a SW package in the toolbox. It is immediately available with its **standard functionality**, but can be modified for special applications according to individual requirements.

Provision is made for two categories of modification:

1. Simple parameterization of standard functionality
2. More fundamental re-configuring of the control unit switchover function

Reasons for more fundamental, user-specific re-configuring (2.) can be as follows:

- Displacement strategy which differs from standard functionality
- Operating mode switchover which differs from standard functionality
- Independent handling of override switch for switchover of control unit
- Existence of a 2nd machine control panel on an MMC/HT6.

---

**Note**

The logic in the MMCs/HT6s (automatic control unit switchover) is fixed. It exists in two variants for the MMC 103.2 and MMC 100.2/HT6. The flexibility of the solution in SW 5 includes the following features:

1. Configuring: NETNAMES.INI
2. See below for parameterization of standard functionality of PLC program “Control unit switchover”.
3. See Chapter 6 for more fundamental user-specific re-configuring.

---

**(1) Standard functionality**

This is implemented as an optional **PLC program**.

**Program structure**

The control unit switchover program consists of:

1. FB101/DB101: Online/offline operating mode switchover
2. FB102/DB102: Active/passive operating mode switchover
3. FC103: Machine control panel switchover

Every program section is implemented in a separate function block (FB) or function (FC). The variables are stored in a separate instance data block (DB) for each FB. The control unit switchover **main program** is stored in **FB101**. The latter block must be called in an organization block (OB) to activate the functions. FB102 and FC103 are called repeatedly in MB101.

**MCP switchover**

The MCP switchover is not mandatory. It can be enabled or disabled via FB101/DB101 variable:
**MSTT_enable**
Assignment in the declaration table:

| | |
|---|---|
| TRUE | MCP switchover is active |
| FALSE | MCP switchover is not active, no FC103 call |

The MCP assigned to the active MMC is always activated. If the operating mode switchover function is disabled, then the MCP assigned to the last MMC to go online is activated.

Power-up condition:

To prevent the previously selected MCP from being activated again when the NCU is restarted, input parameters **MCP1BusAdr** must be set to 255 (address of 1st MCP) and
**MCP1Stop** to TRUE (deactivate 1st MCP) should be set when FB1 is called in OB100.

Enabling commands:

When one MCP is switched over to another, any active feedrate or axis enabling signals may be transferred at the same time.

!

**Important**

After an MCP switchover, the override switch on the new MCP is active immediately.
The keys actuated when the MCPs are switched remain operative. If no MCP is installed on the newly selected MMC, then it will not be possible to cancel the key functions from this MMC. Measures for situations of this type must be implemented in the PLC user program.

**Operating mode switchover**

The operating mode switchover is not mandatory. It can be enabled or disabled via FB101/DB101 variable:
**aktiv_enable**
Assignment in the declaration table:

| | |
|---|---|
| TRUE | Mode switchover is active |
| FALSE | Mode switchover is not active, no FC102 call |

**Error messages**

If disturbances occur (e.g. interface signal failure) while the program is running, corresponding alarms/error messages are transferred to data block DB2. 6 **alarms** are implemented:

1. Error in MMC bus address, MMC bus type
2. No confirmation MMC1 offline
3. MMC 1 is not going offline
4. No confirmation MMC 2 offline
5. MMC 2 is not going offline
6. Online-request MMC is not going online (calling MMC)

and an **error message**:

Sign-of-life monitoring error

When the defaults in FB101 are left unchanged, the alarms begin at
DB2.DBX188.0 (1st alarm) and end at
DB2.DBX188.5. (6th alarm)
With variable:
**DBX_Byte_alarm** the byte value for the 6 alarms can be changed from the default setting of 188. With variable:

**DBX_Byte_report**, the byte value of the operational message can be changed from the default setting of 192.

**Mixed mode**

The term “mixed mode” refers to a state in which a conventional OP without control unit switchover function is connected to the first MMC interface on the NCU. The control unit switchover then operates exclusively on the 2nd MMC interface.

Parameter “MMC_mixed_mode” (variable in FB101/DB101) can be set to switch the operating mode from pure MMC operation to mixed mode.

To ensure that the control unit switchover function operates correctly in mixed mode, the following settings relating to mixed mode must be made:

FB101/DB101 variable:
**MMC_mixed_mode**
The following assignments must be made for an FB101 call:

| | |
|---|---|
| TRUE | Mixed mode is active. Control unit switchover operates on the 2nd interface |
| FALSE | Mixed mode is not active. Control unit switchover operates on both interfaces |

Supplementary conditions:

– A machine control panel (MCP) must not be configured for the first online interface. The first online interface is always assigned **active** mode status in mixed operation.

– To allow the second online interface to be assigned active mode status, it is possible to assign active mode status to both online interfaces in mixed operation. However, certain supplementary conditions apply.

**Warning**

When data are input from both control units simultaneously, there is a risk that inconsistent data will be transferred to the control.

**Server mode**

Once a server has occupied the online interface of an NCU, it cannot be displaced (suppressed) by any other device.

**Processing operating authorization for servers**

Three server-related program branches for handling MMC requests are implemented in the control unit switchover program:

1. Request for relinquishing operating focus
2. Request for setting operating focus
3. Relinquish operating focus

Each branch checks/processes the first and second online interfaces.

The requests are positively acknowledged if no switchover disabling commands are active. The “Relinquish operating focus” includes deactivation of the relevant machine control panel.

See also Figures 6-10 to 6-12.

**Wait times for acknowledgement signals**

To render the program independent of timers, two wait times based on repeated reading of the system time are implemented via SFC64 in the control unit switchover program. The wait times for acknowledgements can be changed if necessary by means of:

FB101/DB101 variables:

| | |
|---|---|
| **waiting_period_1** | Wait for activation/online MMC |
| **waiting_period_2** | Wait for deactivation/offline MMC |
| | Wait for MMC sign of life |

Values of between 0–32 (seconds) can be assigned to the FB101 variables. These values are entered in ms.

**Program integration**

If the control unit switchover program is to be called as a function in a higher-level PLC program, then it must be ensured that FB101, FB102 and FC103 and associated instance data blocks DB101, DB102 have not already been used elsewhere.

**Initialization**

When the NCU is restarted, all signals relating to control unit switchover on the PLC interface in DB19 are set to zero.

---

**Note**

Before an NCU is initialized, it must be ensured that it is **not** currently linked **online** to any MMC/HT6.
It may be necessary to perform an MMC restart.

---

**Resetting of interface by PLC**

The interface signals relating to control unit switchover can be reset selectively as follows (without RESET on the NCU):

FB101/DB101 variable:
**initialization**

| | |
|---|---|
| TRUE | Reset signals in DB19 once. After signal reset the Initialization parameter is automatically reset to FALSE. |

**Signoflife monitoring**

Once an MMC/HT6 is connected online, it sends a sign of life in DB10 DBB108 (separately for both MMC/HT6s). If an MMC/HT6 in online mode does not send a sign of life signal for longer than the time set in waiting_period_2, then the PLC program generates message: “Sign-of-life monitoring error”. This message is not canceled until one of the MMCs/HT6s is switched from offline to online mode again.

**Identifier for MMC/HT6 "Control unit switchover exists"**

In certain operating states, MMCs/HT6s must be able to detect whether the control unit switchover function exists. The "online request" interface signal in DB19.DBW110 **m_to_n_alive** is provided for this purpose. As soon as block FB101 is called in the PLC, it also sends a sign of life signal, consisting of the cyclic incrementation of **m_to_n_alive** (ring counter).

**Generation after adaptations**

After static parameters have been modified in FB101, DB101 must be:
– deleted,
– generated again,
– called and
stored.

**Blocks and functions used**

| | |
|---|---|
| Function blocks | FB101, FB102 |
| Instance data blocks | DB101, DB102 |
| Functions | FC103 |
| DB of interface | DB19 |
| Global data block for error messages | DB2, (DB3) |
| Timer auxiliary function | SFC64 |

## 2.2 Several operator panel fronts and NCUs, standard functionality

The M:N system for all software versions **without** the **Control Unit Management** option is described below. The different performance levels of SW versions from 3.1 onwards are specified in connection with individual functions and as an overview in Chapter 3.

---

**Note**

Section 2.2 does not apply to the HT6, since only one HT6 can be operated on an NCU without control unit management.

---

### 2.2.1 Configurations

**Configuration parameters**

As it is possible to freely combine hardware components, it is necessary to inform the system which components are combined and in what manner. On the MMC 102/103, this is done by means of an operator dialog in the Start-up area. In the case of the MMC 100/OP030, the configuration parameters are entered through the creation of a configuration file which is loaded for start-up. The file must be structured as described below.

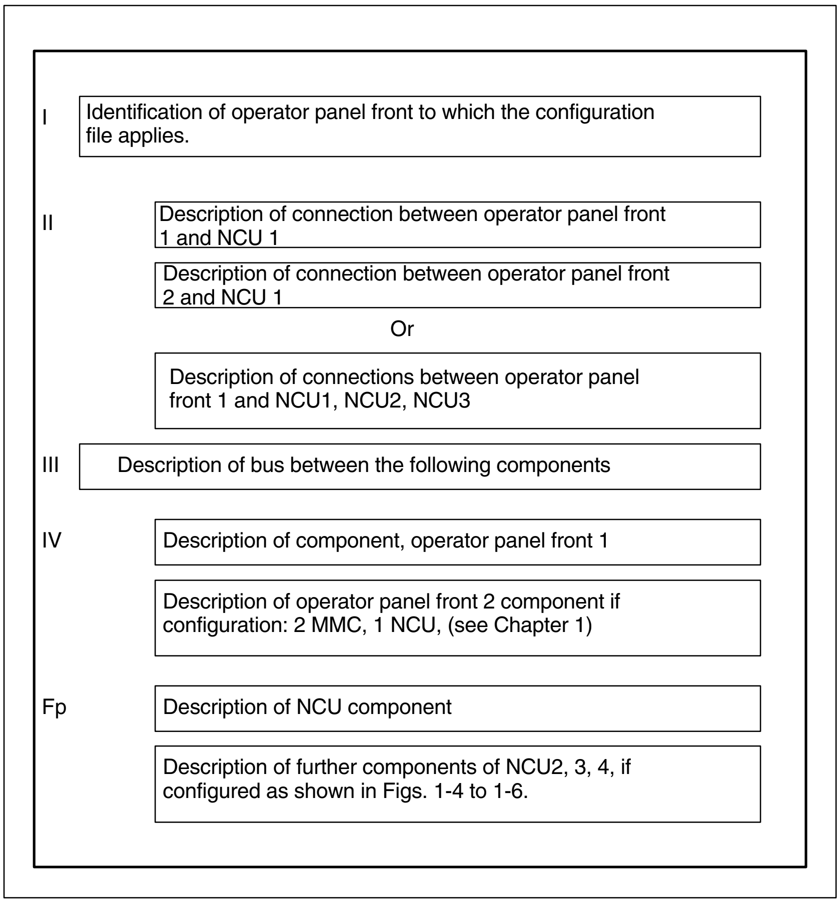



Fig. 2-4 Structure of configuration file NETNAMES.INI

**Examples**

For complete examples of configuration files, please refer to Chapter 6 of this Description.

**Syntactic declarations**

The configuration file must be generated as an ASCII file. The syntax is the same as that used in Windows *.ini" files.

In the following tables, the components which the user may need to adapt or which he can name freely are typed in *italics*. Alternative passwords are specified separated by an I. Passwords must be typed in small letters. Comments can be inserted in the parameter file. They must start with ";" and are limited on the right by the end of line. Blanks may be used as separators at any position except for in identifiers and passwords.

**Number of configuration files**

A configuration file is required for each connected operator panel front. The configuration files of different operator panel fronts included in one configuration differ from one another only in the first entry which contains the assignment of the file to a specific panel front ([own] see below). For practical purposes, the core of the file is generated just once and then copied for the other panel fronts. The identifier of the operator panel front to which the file applies is then inserted in each copy.

**I.**
**Identification of operator panel front**

Identification of operator panel front to which the configuration file applies.

Table 2-2 Identification of operator panel front

| Descriptive entry | Formal | Example |
|---|---|---|
| Header | [own] | [own] |
| Next line | owner = *Identifier* | owner = MMC_2 |

*Identifier* A descriptive entry for an operator panel front must be generated with the selected identifier according to IV.

Vocabulary words:
own Start descriptive entry
owner Owner

**II.**
**Connections**

Description of connections from the operator panel front components to the NCU to be addressed. An entry of the following type is required for each operator panel front.

Table 2-3 Description of connections OP – NCU

| Descriptive entry | Formal | Example |
|---|---|---|
| Header | [conn *Identifier*] | [conn MMC_1] |
| Next line(s) | conn_i = *NCU_ID* | conn_1= NCU_1 |

*Identifier* A descriptive entry for an operator panel front must be generated with the selected identifier according to IV.
*NCU_ID* A descriptive entry for the NCU must be generated with the selected NCU identifier according to V.

Vocabulary words:
conn Start connection entry
conn_i Password for connection (in SW 3.1 only), otherwise i = 1, 2, ..., 8.

**III.**
**Description of bus**

The hardware allows links to be implemented via different buses which are differentiated mainly by their baud rates. The bus type used must be specified.

Table 2-4 Description of bus

| Descriptive entry | Formal | Example |
|---|---|---|
| Header | [param network] | [param network] |
| Next line | bus = opi \| mpi | bus= opi |

Vocabulary words:
param network Start network descriptive entry
bus Bus
btss Operator panel front interface, 1.5 Mbaud
mpi Multi Point interface, 187.5 Kbaud

**Note**

The baud rate is automatically detected on the MMC 100/100.2.

**IV. Description of operator panel front component(s)**

A separate entry must be generated for every individual panel front component connected to the bus. A maximum of two entries in SW 3.x.

Table 2-5 Description of operator panel front component

| Descriptive entry | Formal | Example |
|---|---|---|
| Header | [param *Identifier*] | [param MMC_1] |
| Next lines (optional) | name = *any_name* | name = MMC_A |
| (optional) | type = *mmc_100 \| mmc_102 \| op_030* | type = mmc_100 |
| | mmc_address = j | mmc_address = 1 |

*Identifier* Entry for first or second operator component.
*bel_name* Arbitrary name of max. 32 characters
*mmc_100 \| mmc_102 \| op_030* Type of operator component
*j* Address of operator components on the bus: *j* = 1, 2, ... 15
SW 4.x and higher:= 1, 2, ... 31

Vocabulary words:
param Start parameters for an (MMC) component
name Arbitrary name of operator component to be described
type Type of operator component
mmc_address Bus address of operator component

**V. Description of NCU component(s)**

A separate entry must be generated for every single NCU component connected to the bus.

Table 2-6 Description of NCU component

| Descriptive entry | Formal | Example |
|---|---|---|
| Header | [param *NCU_ID*] | [param NCU_1] |
| Next lines (optional) | name = *any_name* | name = NCU1 |
| (optional) | type = *ncu_570 \| ncu_571 \| ncu_572 \| ncu_573* | type = ncu_572 |
| *) | nck_address = *j* | nck_address = 13 |
| *) | plc_address = *p* | plc_address = 13 |

| | |
|---|---|
| *NCU_ID* | Entry for NCU component. *One* for SW version 3.1 |
| *bel_name* | Any name of max. 32 characters; with MMC 102/103 the name entered here (e.g. NCU1) is output in the alarm line |
| *ncu_570*/ | NCU type, (ncu_570 not applicable to configuration 1 MMC, 3 NCUs) |
| *ncu_571 / ncu_572/ ncu_573* | |
| *j* | Address of NCU component on the bus: *j* = 1, 2, ... 15<br>SW 4.x and higher: = 1, 2, ... 31 *) |
| *p* | Address of PLC component on the bus: *p* = 1, 2, ... 15<br>SW 4.x and higher: = 1, 2, ... 31 *)<br>When bus = btss, *j* and *p* must be set identically.<br>***)** The following applies when bus = mpi:<br>As the associated NCU is always assigned the next-higher address than the PLC, the PLC address must not be 31. address 31 can, for example, be assigned to an MMC. |

Vocabulary words:

| | |
|---|---|
| param | Start parameters for an (NCU) component |
| name | Arbitrary name of operator component to be described |
| type | Type of operator component |
| nck_address | Bus address of NCU |
| plc_address | Bus address of PLC. |

---

**Note**

If the bus node addresses on the MPI bus are configured in conformance with SIMATIC, the configuring engineer can read out the assigned addresses using a SIMATIC programming device and use them to create the NETNAMES.INI file.

---

**Defaults**

The following defaults are applied if no NETNAMES.INI configuring file has been copied into the MMC 100/OP030 or if the file cannot be interpreted:

– The bus type used is automatically determined.
– MMC has address 1.
– OP030 has address 10.
– NCU and PLC both have address 13 for an OPI bus.
– For an MPI bus, the NCU has address 13 (SW 3.5 and later: **3**) and the PLC address 2.

If the network configuration actually corresponds to these default settings, then it is not necessary to explicitly generate and load a NETNAMES.INI file. If, however, a special file is generated, then it must correspond exactly to the actual network on account of the special features described below.

**Compatibility**

The use of the above defaults establishes compatibility with earlier software versions for operation of the panel front.

## 2.2.2 Switchover of connection to another NCU (SW 3.2 to 3.x)

**User interface**

The data area menu has been extended by the soft key “Connections”. This goes to a submenu in which the connections (conn_1, ... conn_n) declared in NETNAMES.INI are displayed for selection via individual soft keys. The name (name=...) allocated to the connection in NETNAMES.INI is displayed on the soft keys. A connection to the new NCU is established by actuating the corresponding soft key.

In detail, the behavior depends on the type of MMC.

**Changeover behavior on OP030**

ONLINE change to another bus node is not possible on the OP030. The NETNAMES.INI file contains a permanently configured connection.

**Changeover behavior MMC 100**

The soft key “Connections” is only displayed if more than one connection is implemented in NETNAMES.INI. When changing to the new NCU, the existing connection to another NCU is interrupted. MMC applications, at the instant of link changeover, must no longer need the link to the previous NCU (e.g. for active data backup via RS-232 interface). If this rule is contravened, the control system outputs a corresponding message.
Concerning the NCU to which the changeover takes place, the MMC behaves as with a restart. In this case, it is in the operating area set as the “start operating area”.

**Changeover behavior MMC 102/103**

The “Connections” softkey is only displayed if the M:N function is activated on the control. The “M:N” function is activated in the “Start-up/MMC/Operator panel front” menu. All communications connections remain established with any changeover and the applications which have used these connections remain active. Concerning the new NCU, the MMC is after the changeover in the same operating range as before with another NCU.

**Possible defects**

With change of the connection to another NCU, it is possible that the NCU with which the connection is to be established rejects this. There may be a defect in the NCU or no other MMC unit can be operated at that time from the NCU any more.
MD 10134: MM_NUM_MMC_UNITS (number of possible simultaneous MMC communications partners) contains the setting which defines how many MMCs can be processed by an NCU at one time. The OP030 uses one unit, the MMC 100 and MMC 102/103, as supplied, each use two units. More units (up to 12) are required for larger OEM packages.

**Alarms/messages**

MMC 100, OP030
Only the alarms of the NCU with which a link is currently active can be output. Acceptance of configuration acc. configuration diagram in Chapter 1, subsection “Configurability”.

MMC 102/103
The alarms and messages of all connected NCUs can be processed simultaneously.

**Alarm text management**

MMC 100, OP030
Only one version of the alarm texts can be stored on the operator component. The standard alarm texts are stored once in the same formulation for all NCUs. The possible alarms for all connected NCUs must be stored in the one possible area for user alarms.

MMC 102/103
It is not possible to set up user alarm texts that apply specifically to the NCU (MMC only manages one alarm text file, SW 3.4 MMC 102/103).

**Link check**

MMC 100, MMC 102/103

The address of a connected NCU (on OPI bus only) can be altered in the “Connections/Service” menu.
The new NCU address is stored on the NCU.

The soft key labeled “Service” is only displayed if the password for “Protection level service” has been entered.

When the function is started up, a direct connection between the MMC and the relevant NCU must be established before the address is altered to ensure that the address is not programmed more than once on the bus.
(See paragraph “Power-up” below for instructions on modifying the address.)

---

**Note**

With replacement of the NCU (service case) or with failure of the backup battery, the address is no longer stored.
A general reset on the NCU does not delete the NCU address. The address can only be changed via an MMC.
To ensure that the current connection is shown in the basic display, the channel name must be assigned unambiguously in MD 20000: CHAN_NAME (channel name).

---

### 2.2.3 Switchover of connection to another NCU (SW 4 and higher)

---

**Note**

The channel menu function is an option and must be configured in the “NETNAMES.INI” file.

---

You can change to the channel menu in all operating areas by activating the channel switchover key. In this case, only the horizontal and vertical softkeys change.
The horizontal softkeys are for selecting a channel group (max. 24), up to 8 connections to channels in different NCUs can be set up in one channel group.

The “Channel menu” screen displays all current communication connections and the associated symbol names.

### 2.2.4 Creating and using the configuration file

**MMC 100, OP030**

The NETNAMES.INI ASCII file generated on the PC or programming device is loaded, as described in

**References:** /IK/, Installation Kit

via the RS-232 interface and permanently stored in the FLASH memory of the control units.

**MMC 102/103**

The NETNAMES.INI file can be processed directly with an editor (in menu "Start-up/MMC/Editor" or DOS_SHELL) on the hard disk of the operator component. The NETNAMES.INI file is stored in the installation directory:
C:\MMC2
SW 4 and higher C:\USER\NETNAMES.INI.

### 2.2.5 Booting

**Differences between MMC 100 and MMC 102/103**

Owing to the differences in operating and power-up characteristics, different start-up procedures are required.

– MMC 100 always runs in "M:N" mode, when "M:N" is configured in the NETNAMES.INI file.

– The mode can be set in the "Start-up/MMC/Operator panel front" menu on the MMC 102/103. The MMC 102/103 always runs in a "1 : 1" link with an NCU, the NCU address can be specified directly. If the "M:N" mode is set on the MMC 102/103, then the MMC searches the NETNAMES.INI file for the names of the partners specified for this function. The addresses are freely assignable.
**Recommendation**: Keep address 0 free (for PG )
Keep address 13 free (for servicing: NCU replacement)

– The OP030 is not functionally capable of "M:N". It can be used as a second operator panel front that is permanently assigned to an NCU ("1 : 1" link). The addresses of the connected partners can be set for this purpose.

---

**Note**

It is advisable to make a written record of the procedure (address assignments, etc.) beforehand.

---

**Installation and startup**

The NCUs are assigned bus address 13 in the delivery state. Every NCU on the bus must be allocated its own, unique bus address.

Addresses are assigned in:

– MMC: NETNAMES.INI file
– NCK: "Start-up/NC/NCK address" menu
– MCP: Switch... (address and possibly baud rate, see also /IBN/) OB100 parameters: ...(see also FB1/P3).

---

**Note**

An NCK address is not deleted with "Delete SRAM" (switch S3= position "1" on NCU).

---

**Power-up with MMC 100**

The power-up process is the same as described in Subsection 2.1.6 for the MMC 100.2.

**Power-up with MMC 102/103**

The power-up process is the same as described in Subsection 2.1.6 for the MMC 102/103.

## 2.2.6 NCU replacement

In the case of NCU replacement or an additional NCU, the procedure is analogous to start-up (see 2.2.5).

**Variant 1**

1. Establish 1:1 connection between MMC and NCU
2. Power-up MMC on NCU with bus address "13" (see above)
3. Enter new NC address via the Start-up/NC/NC address operating area and boot NCU.
4. Wire bus again for M:N operation

**Variant 2**

1. The NCU, which is the "power-up NCU" for an MMC connected to the bus, is disabled. (The MMC powers up at the first connection configured in the NETNAMES.INI file)
2. Power-up MMC on NCU with bus address 13 (see above)
3. Enter new NC address via the Start-up/NC/NC address operating area and boot NCU.
4. Activating "Power-up NCU" again

---

**Note**

– Bus address 13 must be reserved for servicing purposes (i.e. must not be assigned to a bus node).
– MMC 100/100.2:
The name length in file NETNAMES.INI (configuring in channel menu) is limited to 5 characters.
– MMC 102/103:
The data "mst_address" is not evaluated, but used for the purpose of bus node documentation.
If the channels are on different NCUs, "m:n" must be set in the operating area Start-up/MMC/Operator panel front.

---

**Data exchange between NC<->PLC**

In configurations consisting of 1 x MMC and n x NCU, it is often necessary to synchronize the NCs.

The following synchronization options are available:

– NCK I/Os on drive bus (digital, analog, writing of NC and PLC).

– Normal PLC I/Os (I/O link).

– Link via PROFIBUS-DP (PLC-CPU315 required).

– Link via the global data function of SIMATIC S7 (PLC-CPU315 required). This option is also available on the PLC-CPU 314 with SW 3.5 and higher.

## 2.3 Restrictions in relation to equipment

**Rejection of link**

On switchover to another NCU, the NCU selected for the new link may reject the connection. The cause may be a defect in the NCU or no additional MMC unit can be accepted. In this case, the MMC 100 automatically switches over to connection 1 after approx. five seconds. MMC 102/103 displays “#“ for the variables.

**Alarms, messages**

Handling of alarms/messages is dependent on the MMC type:

1. MMC 100/OP 030
   Due to the equipment restrictions on driver level and the limited working memory, alarms/messages of only one NCU can be processed simultaneously.
2. MMC 102/103
   The MMC manages only one alarm text file. The NCU name assigned in the NETNAMES.INI file is displayed as the NCU identifier in front of every alarm or message. To obtain user texts specific to the NCU, it is possible to define user areas in the PLC for certain NCUs. The alarms/messages of all connected components can be processed and displayed simultaneously.

**Operator interface**

Handling of alarms/messages is dependent on the MMC type:

1. MMC 100
   Fields and variables of one NCU can be displayed simultaneously in a window. Alarms and messages are displayed only by the NCU that is currently linked to the MMC.
   Up to four connections (one active connection (alarms, messages), three other connections) can be displayed simultaneously via user configuration (OEM), whereby all variables of a connection must be contained in one window (window-specific connections).
2. MMC 102/103
   Generally, fields and variables of different NCUs can be displayed in the same window (as OEM application).
   Alarms and messages can be displayed on all NCUs (to which the MMC has a connection).
3. OP030
   OP030 can only be configured as a “1 : 1” connection to an NCU.

When the MMC 100 and MMC 102/103 are used in the standard configuration (Chapter 1, subsection “Configurability”), it is not necessary to configure the operator interface. If variables of different NCUs must be output simultaneously in a display, configuration is necessary.

**References:** /PK/, SINUMERIK MMC 100/EBF Configuration kit

With MMC 100, all variables of a window must proceed from one NCU. With MMC 102/103, a suitable mixing of the variables of different NCUs is permissible.

## 2.4 NCU link

### 2.4.1 Introduction

Owing to the limitation on the memory and computing capacity elements, the number of channels or axes per NCU is restricted. A single NCU is not sufficient to fulfill the requirements made by complex and distributed machines, such as multi-spindle and rotary indexing machines. For this reason, the control system and closed-loop axis controls are distributed among several NCUs.

In order to ensure, however, that channels and axes can continue to operate on an interrelated, cross-NCU basis, the system provides so-called

- NCU link functionality.

This includes:

**Functional expansions**

The following applications are possible in SW 5 and higher:

- Cross-NCU interpolation (coupling of setpoints, actual values and VDI signals)
- Real exchange of axes
- Cross-NCU access to axis values and axial system variables
- NCU-user communication supported by NCU link variables
- Generation of alarms on the NCU affected by an irregularity, even if the cause of the problem is on another NCU.

## 2.4.2 Technological description

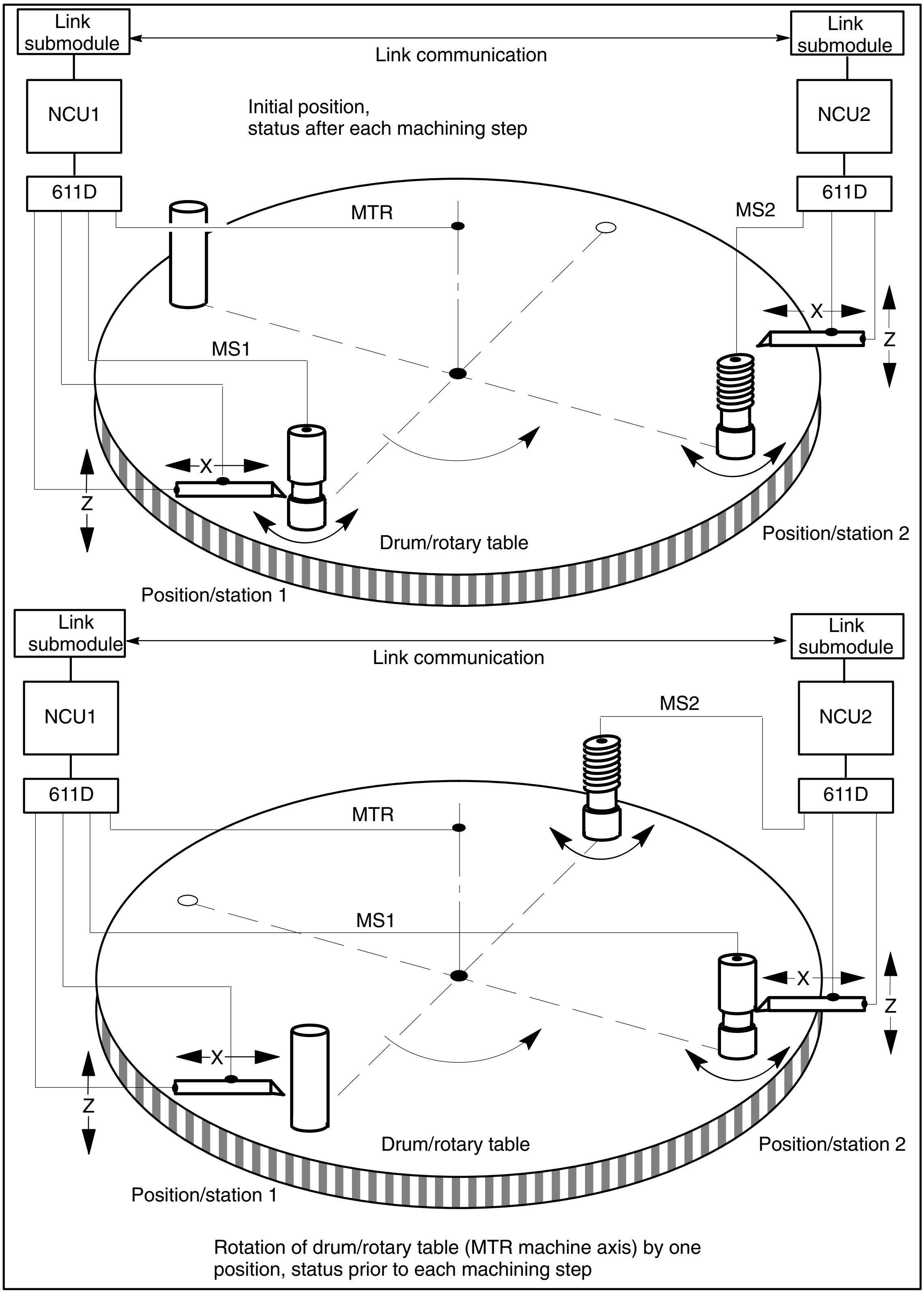



Fig. 2-5 Sectional diagram of a drum changeover

Fig. 2-5 shows the main components of a simple multi-spindle plant. Several spindles are mounted mechanically on the drum, each of which can used to perform a different machining operation. Together with the slide (X and Z axes), they form a machining station which is assigned to one channel. A workpiece is rotated by one spindle.

The workpiece to be machined is loaded and unloaded only once. The tool is mounted on the slide (e.g. X, Z axes). Various different tools can be loaded for each machining operation.

The tool is continuously assigned to the machining station. The workpiece-holding spindles are moved from one machining station to the next.

The spindle can only be checked for the current machining process. The channel must be able to address the slide axes and the current spindle at any given time. Every time a spindle moves on to the next machining unit, however, the spindle addressed by the channel must be a different machine axis. The "**axis container**" concept solves the variable imaging of channel axes on machine axes. The machine axes might belong to another NCU connected by means of the NCU link. An accessible machine axis belonging to another NCU is referred to as a **link axis** (see 2.5).

The following subjects are closely related to the NCU link function and dealt with in separate subsections.

- Link axes
- Axis container
- User communication across the NCUs
- Configuration of the link grouping

# 2.5 Link axes

---

**Note**

NCU link is available in conjunction with MMC 103.
The corresponding option is required to be able to define the number of available link axes.

---

**Introduction**

This subsection describes how an axis (for example, B1 in Fig. 2-6), which is physically connected to the drive control system of NCU2, can be addressed not only by NCU2, but also by NCU1.

**Prerequisites**

- The participating NCUs, NCU1 and NCU2, must be connected by means of high-speed communication via the link module.

**References:** /PHD/, Configuring Manual NCU 571-573.2, Link Module

- The axis must be configured appropriately by machine data.
- The link axis option must be installed.
- Link communication must be activated with MD 18780: $MN_MM_NCU_LINK_MASK. The link grouping must be configured as described in 2.5.1.

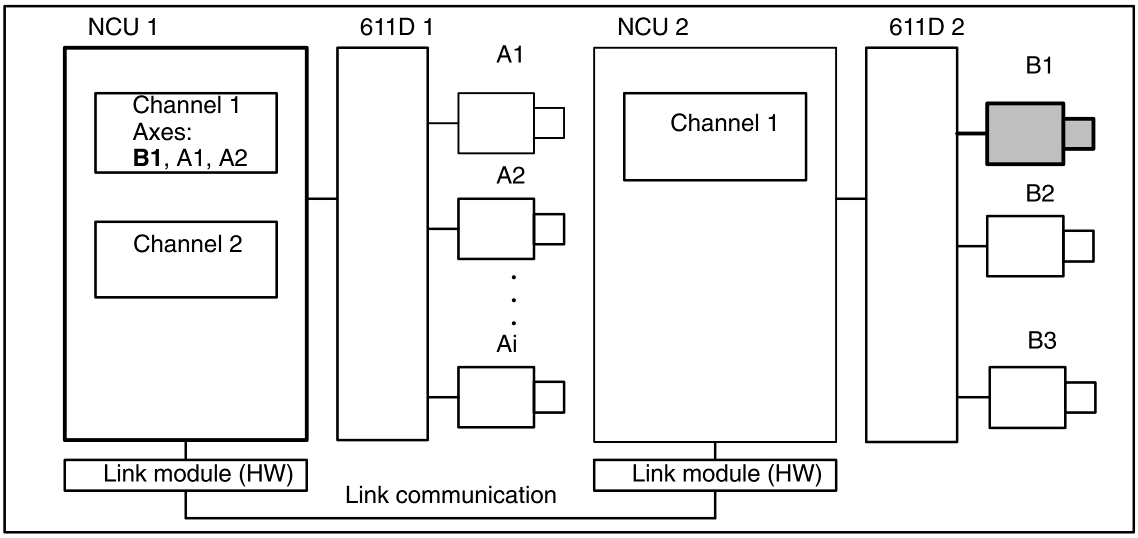



Fig. 2-6 Overview of link axes

**Terms**

The following terms are important for understanding the subsequent description:

- **Link axis**
  Link axes are axes that are physically connected to another NCU and controlled by their servo loop. Link axes can be assigned dynamically to channels of **another** NCU. From the standpoint of a particular NCU, they are not –> local axes. Dynamic changes in the assignment to a channel (exception: channel on another NCU) are implemented according to the 2.6 **Axis container** concept described above.
  Axis exchange with GET and RELEASE from the parts program is only available for link axes within an NCU. In order to cross the NCU limit, the axis must first be placed in the NCU or a channel using the axis container function so that it can then be exchanged optionally in the same way as any other axis.
- **Local axis**
  A local axis is only addressed by the NCU to whose drive bus it is connected.
- **Link communication**
  The link communication is implemented by link modules on the NCUs involved. The link communication consists of setpoints, actual values, alarm handling, global variables (data) and signals (axis signals, PLC signals).
- **Home NCU**
  The NCU which establishes the drive bus connection for a –> link axis and implements the position control is called the home NCU of the link axis.
  In Fig. 2-6 NCU2 is the home NCU for –> link axis B1.
- **Interpolation**
  The **Link axis** option enables for NCUs with –> Link communication interpolation between –> local axes and axes on other NCUs.
  If the interpolation is not only local, cyclical data exchange (setpoints, actual values, ...) takes place within an interpolation cycle. In particular, this causes dead time when waiting for external events.
- **Axis change**
  Use of a –> link axis by a specific NCU can change dynamically. An **axis container** mechanism is provided for this purpose as described in 2.6. The parts program command GET is not available for link axes; the parts program command GETD is only available within an NCU..
  Up to SW 4, it was only possible to exchange axes between different channels of an NCU.
- **Configuration of link axes**
  NCUs that want to use the –> link axes must configure the **NCU identifiers** for the home NCU of the link axis in addition to the usual channel and axis machine data.
- **Home channel**
  Channel in which the setpoint-generating parts program for the axis is executed after the installation has powered up.
- **Lead link axis from SW 6**
  From the point of view of NCU (2) that traverses following axes, a *leading axis* that is traversed by another NCU (1). The required data for the master value axis are supplied via –> link communication for NCU (2). Axis linking between the leading axis and the following axis/axes is implemented, for example, by means of a curve table.

---

**Note**

Link axes expand the limit set by the number of possible connections on the drive bus: With a maximum of 16 NCUs on the NCU link there are theoretical limits of max. 160 channels and 4960 axes/spindles. The following maximum values apply for one NCU: 31 simultaneous axes from 31 local and 32 link axes.

---

## 2.5.1 Configuration of link axes

**SW 4**

Up to SW 4, channel axes are directly mapped on the machine axes of the same NCU via MD 20070: AXCONF_MACHAX_USED (see Fig. 2-7 left).

**SW 5**

From SW 5, the channels operate with one of 31 logical axes from the logical machine axis image. This image points to:

- local axes
- Link axes
- Container slots.

Container slots in turn point to:

- local axes or
- link axes.

The following diagram illustrates the interrelationships:

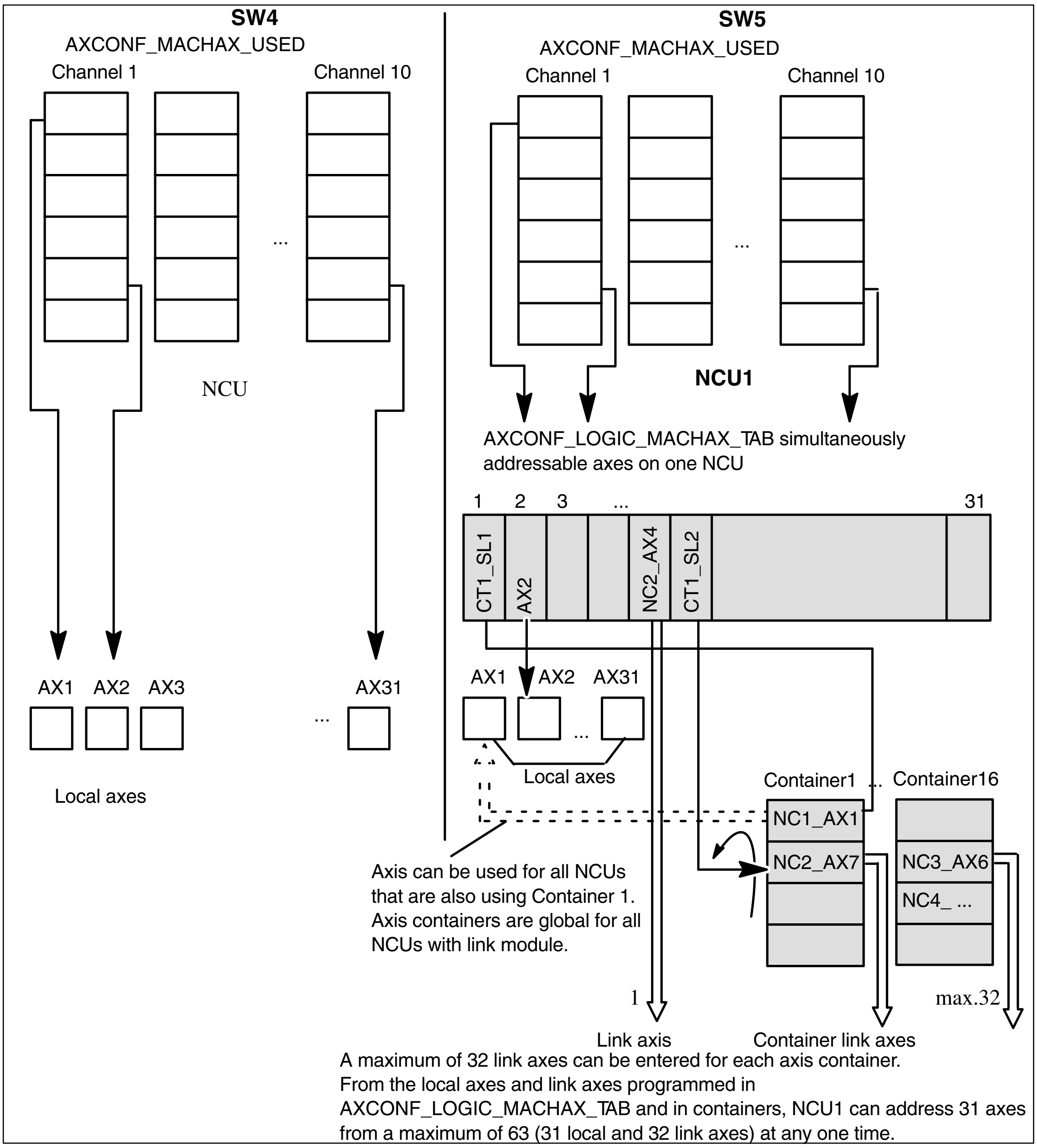



Fig. 2-7 A schematic comparison of the SW4 and SW5 configuration

**With link axes**

To enable link axes to be addressed throughout the system, the configuration must contain information about the axis NCUs. There are two types of NCU axis, i.e. local axes and link axes.

**Differentiation local/link axes**

The table that must be created by means of MD 10002:
AXCONF_LOGIC_MACHAX_TAB
differentiates between local and link axes. See Fig. 2-7 center right and Fig. 2-8.

---

**Note**

The **axis container** functions are described in Section 2.6.

---

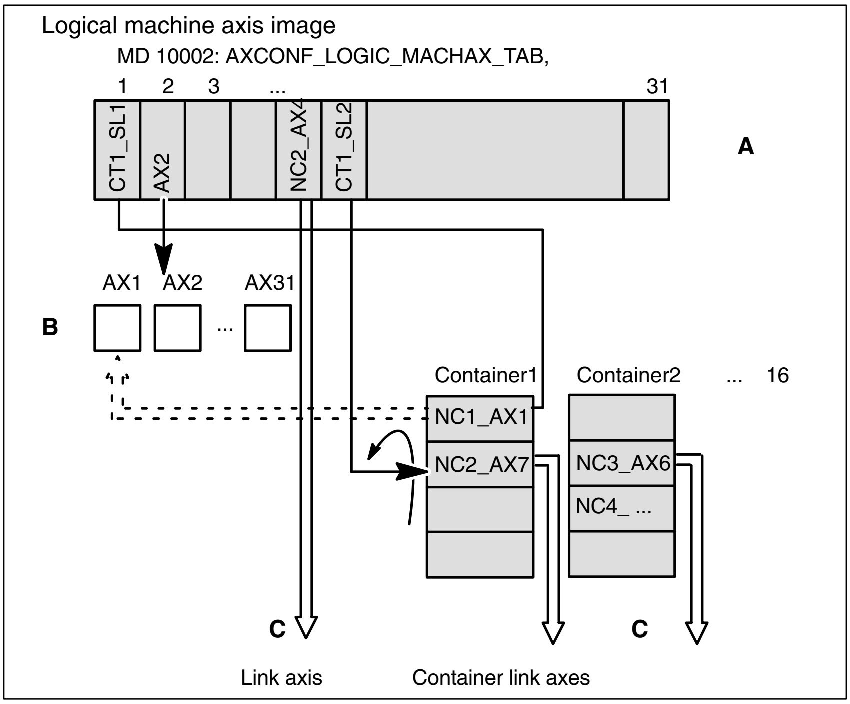



Fig. 2-8 Assignment of channel axes to local machine axes and link axes

## Explanation

The **logical machine axis image** A addresses local machine axes B and link axes C.

The number of local machine axes in B is limited. The maximum permissible number for a specific system can be found in Catalog NC60.1.

All axes that can address the NCU are contained in B and C together.

**Entries in A** have the following format:

$MN_AXCONF_LOGIC_MACHAX_TAB[n] = NCj_AXi
where

| | |
|---|---|
| n | index in Table A |
| NC | stands for **NCU** with |
| j | **NCU number**, 1 <= j <= 16 |
| i | axis number, 1 <= i <= 31 |

Channel axes are no longer directly assigned to machine axes in MD 20070: AXCONF_MACHAX_USED as they were in SW version 4 and earlier, but are now assigned to **logical machine axis image A**.

Viewed from the parts program, the only accessible machine axes are those which can be addressed by the channel (possibly via axis container, see below) via the logical machine axis image at a given point in time.

**Default**

By default, the settings of logical machine axis image A are
local axis name AX1 for entry 1,
and local axis name AX2 for entry 2,
... .
With these, MD blocks that were generated for SW lower than 4 can still be used with SW 5, if only local axes are addressed.

**Examples**

For example, the logical machine image can contain the following expressions:

| | |
|---|---|
| NC2_AX7 | Machine axis 7 of NCU 2 |
| AX2 | Local machine axis 2 |

If only expressions of the latter format AXi are entered in the logical machine axis image, this corresponds to a configuration up to Software Version 4, where only local axes are addressed.
Caution: The default settings are as follows:
MD 10002: AXCONF_LOGIC_MACHAX_TAB[0] = AX1
MD 10002: AXCONF_LOGIC_MACHAX_TAB[1] = AX2
...

---

**Note**

Another valid format for entries in the logical machine axis image A is:
MD 10002: AXCONF_LOGIC_MACHAX_TAB[n] = CTx_SLy where

| | |
|---|---|
| CT | stands for **container** |
| x | container number, 1 <= x <= 16 |
| SL | stands for **Slot** |
| y | slot number, 1 <= y <= 32 |

**Axis containers** represent a grouping of axes which can be altered dynamically. Axis containers are described in Section 2.6.

---

## 2.5.2 Axis data and signals

**Introduction**

Axis data and signals for a link axis are produced on its home NCU. The NCU that has caused the movement of a link axis is provided with axis data and signals from the system:

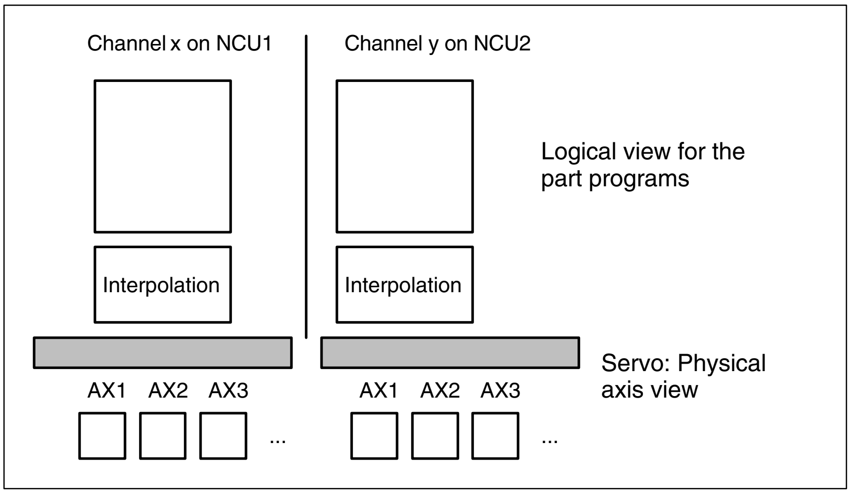



Fig. 2-9 Views of axes

**Implicitly active link communication**

During interpolation, data are made available for axes which are physically subordinate to a non-local servo (identifiable from entries in MD10002: AXCONF_LOGIC_MACHAX_TAB or axis container) via the link communication in the same manner as they are provided for local axes from the logical viewpoint of parts programs. The procedure remains concealed from the applications.

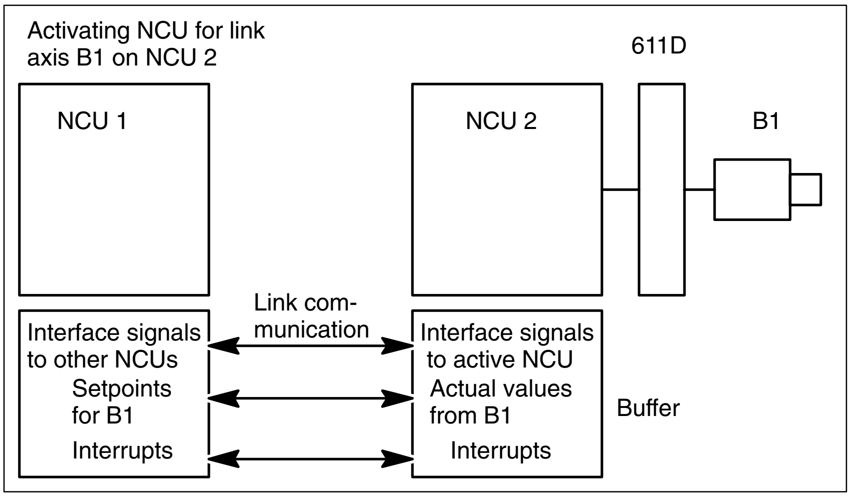



Fig. 2-10 Exchange of operating data and signals of a link axis

**Position control**

The servo loop is implemented on the NCU on which the axis is physically connected to the drive. This NCU also contains the associated axis interface. The position setpoints for link axes are generated on the active NCU and transferred via the NCU link.

**Communication methods**

There are two types of link communication:

- Cyclic communication
- Non-cyclical communication

**Cyclical communication**

– Setpoint for the link axis,
– Actual values from the link axis
– Status signals of the link axis
– Status signals of the NCUs
are transmitted cyclically. Actual values and status signals of a link axis are updated and made available to the NCU that is interpolating this axis.

**Non-cyclical communication**

– Exchange of link variables
– Warm start requirements
– Activation of axis container rotation
– Modifications to NCU-global machine and setting data.
– Activation of axial machine data for link axes
– Alarms

**Transfer time**

Delays incurred for transferring **setpoints** to the home NCU of a link axis and returning its **actual values**. With an interpolation group of local axes and link axes, the control delays the setpoints for the local axes of the interpolating NCU by one interpolation cycle, such that consistent values are taken into account for the interpolation.

If a channel needs the actual values of an axis of another NCU, e.g. a spindle with thread cutting, two interpolation cycles will lapse before they are available. The setpoints then generated are sent one interpolation cycle later to the position control for the above reason.

**Response of the AXIS-VAR server to errors**

If the server cannot supply any values for an axis (e.g. because the axis concerned is a link axis), then it returns a default value (generally 0).

For the purpose of testing, machine data 11398:
AXIS_VAR_SERVER_SENSITIVE can be used to set the axis data server sensitively so that it returns an error message rather than default values.
0: Default value
1: Error message

### 2.5.3 Supplementary conditions for link axes

**Output of alarms from position controller or drive**

Axis alarms are always output on the NCU which is producing the interpolation value. If an alarm is generated for a link axis by the position controller, then the alarm is transferred to the NCU which is currently processing the interpolation.

On the assumption that axis alarms which cause the NCK-Ready relay to drop out (Nck-NoReady) are attributable to faults on the drive bus, the alarm is also output on the NCU to which the axis or the drive bus is physically connected. The reaction “Ready relay dropout” is only activated on this NCU.

**Output of alarms following EMERGENCY STOP**

If an EMERGENCY STOP request is activated by the PLC on an NCU, then all axes physically connected to drives on this NCU are switched to follow-up mode. This means that even axes which are being interpolated by a different NCU are also switched to follow-up. Since this status prevents any further constructive machining operations on the other NCUs, an additional alarm is generated which is designed to stop all axis motions instantaneously.

This additional alarm must be acknowledged by an operator panel reset. If the original (EMERGENCY STOP) alarm is still active at this time, then the additional alarm can be successfully reset, but another alarm (self-clearing) is then produced which prevents axis motion or a new program start until the original alarm has been acknowledged.

**Output of alarms with alarm reaction NCK-NoReady**

If a serious alarm resulting in dropout of the NCK-Ready relay is activated on an NCU, then the effects of the alarm will apply to all other NCUs which are addressing an axis via link communication on the first NCU. An additional alarm which causes all other axes to stop instantaneously is activated on each of the other NCUs.

For alarm acknowledgement, see EMERGENCY STOP.

**Compensation**

The compensation functions

– CEC
– EEC
– QEC

are **not** available for link axes.

**Switching off grouped NCUs**

If an NCU assigned to an NCU grouping is switched off or restarted by NCK RESET, then the other NCUs in the grouping are also affected (see also 2.8). An alarm is generated on the NCUs which are still running to prevent them continuing with the machining operation.

**Powering up an NCU grouping**

If one NCU in the grouping is restarted, e.g. due to changes to machine data, then the other NCUs in the grouping also execute a warm restart.

**Nibbling and punching**

To execute nibbling and punching operations, high-speed inputs and outputs must be connected and parameterized on the “interpolation” NCU (on which the parts program is being executed). Commands “High-speed nibbling and punching”, e.g. PONS and SONS are not available for link axes.

**Travel to fixed stop**

If an axis container axis is being held against a fixed stop, the axis container cannot rotate. Axes can travel to fixed stops on different NCUs and be subsequently clamped without restriction.

**Frames**

Link axes may be included in the program commands for frames only if they are geometry axes as well. The command only changes the geometry for the channel in which the axis is currently assigned. A frame command for an axis which is not defined as a geometry axis is rejected with alarm 14092.

**Revolutional feedrate**

Although setting data 43300: ASSIGN_FEED_PER_REV_SOURCE referred directly to a machine axis in SW 4 and earlier, the MD refers in SW 5 and higher to the logical machine image and, via this, to a machine axis (local or link axis).

### 2.5.4 Programming with channel and machine axis identifiers

**Channel axis identifiers**

Example:

```
WHENEVER $AA_IW[Z] < 10 DO ...          ;Current position of Z axis
```

**Machine axis identifiers**

Example:

```
WHENEVER $AA_IW[AX3] < 10 DO ...        ;Scan current position of
                                        machine axis AX3
```

This method of programming is permitted only if machine axis AX3 is known in the channel at the time of scanning.

---

**Note**

In SW 5.2 and higher, system variables which can be used in conjunction with channel axis identifiers are specially marked in the Advanced Programming Guide (Appendix).

---

### 2.5.5 Flexible configuration

**Introduction**

Rotary indexing machines and multi-spindle machines have special requirements as regards the flexible assignment of channel axes to machine axes.

**Requirement profile**

When advancing the table of the rotary indexing machine or the drum of the multi-spindle machines the axes/spindles are brought to a new station or position. The NCU which controls the axes of a station as local axes must be able to address the newly changed axes/spindles. The hitherto addressable axes/spindles can now be discarded for this purpose.

**Solution**

A configuration of the relevant axes in an axis container specified in machine data enables different machine axes to be located in succession behind a channel axis that remains constant. Advancing the rotary table or drum is performed synchronously with the advancing of the axes entered in the axis container.

Axes in an axis container can also be configured as geometry axes.

---

**Note**

The axis container has no mode group reference, i.e. the workpiece-holding, traveling axis can change from one mode group to another at different machining stations.

---

## 2.6 Axis container

**Definition**

An axis container can be imagined as a circular buffer in which

- local axes and/or
- link axes

are assigned to channels. Axes in an axis container are also referred to as **container axes**. Assignments can be "shifted" ("rotation" of the circular buffer) by means of program commands. The term axis in this case refers to both axes and spindles. All machine axes in the axis container must be assigned to exactly **one** channel axis at any given point in time.

---

**Note**

Rotation of the drum or rotary table is analogous to the rotation of the circular buffer with the assigned axis entries.

---

**Description**

The link axis configuration described in Subsection 2.5.1 allows reference to be made to axis containers in the logical machine axis image, in addition to direct reference to local axes or link axes. This type of reference consists of:

- a container number and
- a slot (circular buffer location within the container)

The entry in a circular buffer location contains:

- a local axis **or**
- a link axis

(either axis or spindle)

**Axis container names in SW 5.2 and higher**

Axis container names can be freely defined with machine data
MD 12750 : AXCT_NAME_TAB
in SW 5.2 and higher. The names assigned can then be used:

- in axis container rotation commands AXCTSWE( ) and AXCTSWED( ) to address the container to be rotated and
- when scanning the states of axis containers using system variables:
  – $AC_AXCTSWA[ ]
  – $AN_AXCTSWA[ ]
  – $AN_AXCTAS[ ]

**SW 5.1 parameter CTi**

In this software version, **channel axis identifiers**, which refer to the container to be rotated via the logical machine axis image, must be used instead of axis container identifiers.

**Definition of container contents**

Machine data MD 12701 ... 12716: $MN_AXCT_AXCONF_ASSIGN_TAB1 ...n defines the default assignment between an axis container slot and a machine axis within an NCU grouping for axis container 1...n. The assignment between an axis container slot and the selected channel is programmed in MD 20070: $MC_AXCONF_MACHAX_USED and MD 10002: $MN_AXCONF_LOGIC_MACHAX_TAB.

**Example**

In the example in Fig. 2-11, the 3rd channel axis (3rd entry in $MC_AXCONF_MACHAX_USED) is a container axis. The 3rd entry in $MC_AXCONF_MACHAX_USED refers to the 8th entry in $MN_AXCONF_LOGIC_MACHAX_TAB and this (CT3_SL2) in turn to the 3rd axis container and its container slot 2. This 2nd entry in $MN_AXCT_AXCONF_ASSIGN_TAB3 (NC3_AX1) defines the 1st machine axis of NCU3 as a container axis of axis container 3, i.e. in the initial state, the 4th channel axis is the 1st machine axis of NCU3.

The 5th channel axis is also a container axis: The 5th entry in $MC_AXCONF_MACHAX_USED refers to the 7th entry in $MN_AXCONF_LOGIC_MACHAX_TAB and this (CT1_SL1) in turn to the 1st axis container and its container slot 1. This 1st entry in $MN_AXCT_AXCONF_ASSIGN_TAB1 (NC1_AX1) assigns the 1st machine axis of NCU1 to the 1st slot of axis container 1, i.e. in the initial state, the 1st machine axis of NCU1 is assigned to the 5th channel axis.

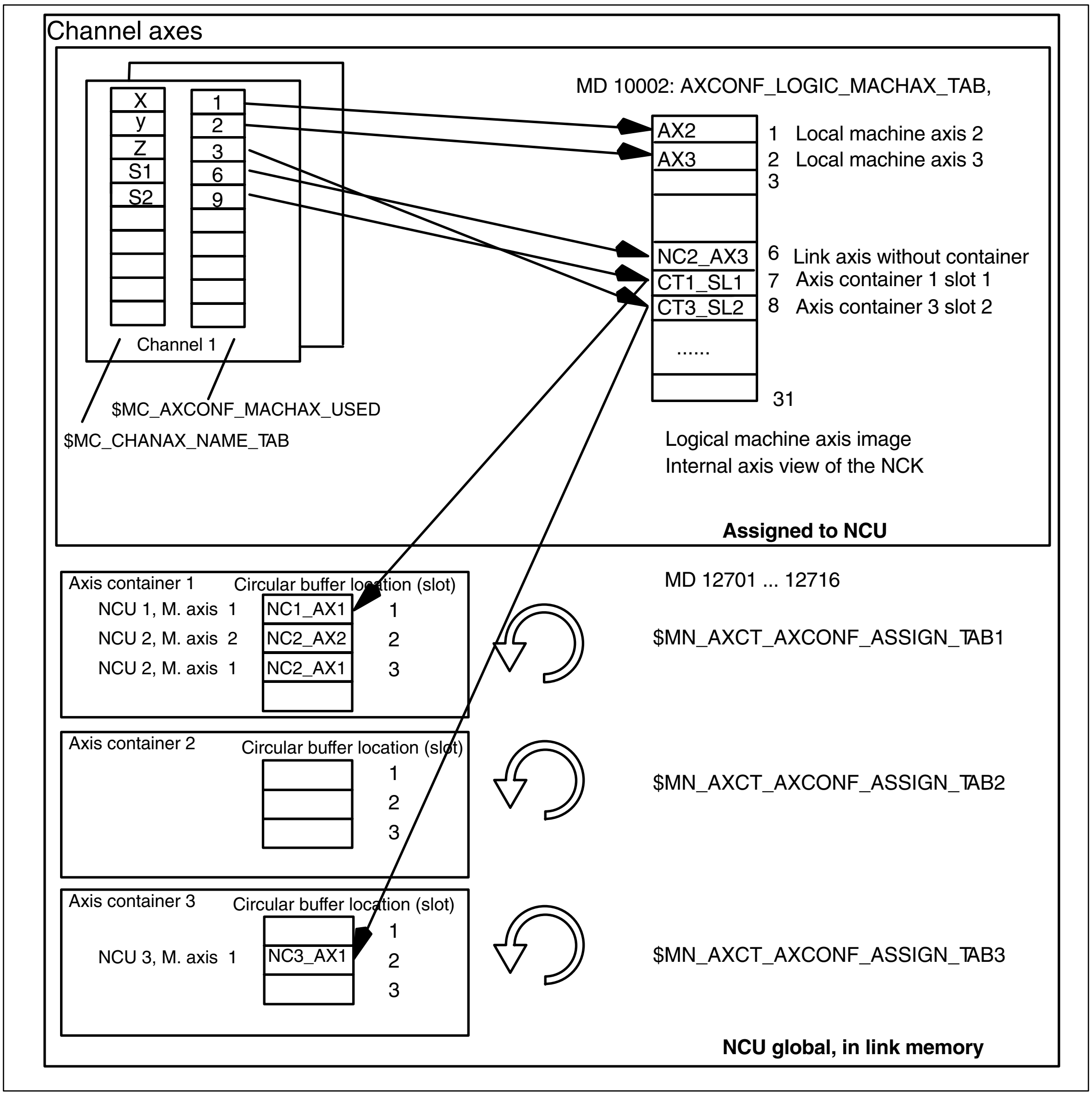



Fig. 2-11 Mapping of channel axes onto axis containers via logical machine axis image

Axis container entries contain local machine axes or link axes from the perspective of an individual NCU. The entries in the logical machine axis image MD10002: AXCONF_LOGIC_MACHAX_TAB for a single NCU are permanent.

**Container rotation**

The contents of the axis container slots are variable inasmuch as the contents of the circular buffer (axis container) can be shifted together by $\pm$ n increments. The number of increments n is defined for each axis container in
SD 41700: $SN_AXCT_SWWIDTH.
The number of increments n is evaluated modulo in relation to the number of actually occupied container slots. In doing so, new contents are created for all slots of an axis container (exception: 0 and slot number = increment number).

System variables provide information about the current status of an axis container; these system variables can be read addressed from the parts program and synchronized actions. See 2.6.1.

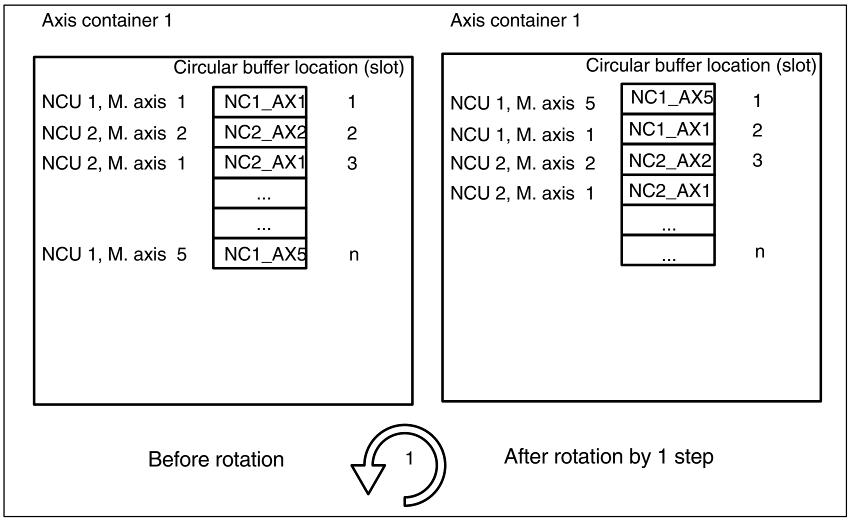



Fig. 2-12 Shifting the entries to the axis container slots

The axis container model has the following characteristics:

- A channel always sees a fixed number of axes with defined channel axis names (logical machine axis image)
- The “rotation” of the axis container sets new machine axes for **all** channels that have axes in the same axis container.

**Frames and axis container rotations**

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

**Activation of axis container rotation**

The application must ensure that the desired local or link axes are addressed by issuing commands in the parts program for rotating the axis container to a specific position.

For example, when rotating the drum of a multi-spindle machine into a new position, it must be ensured that each position addresses the newly changed spindle by rotation of the axis container.

---

**Note**

Axis containers can be used jointly by different channels of an NCU and by channels of other NCUs.
If axes of different channels display reference to the same axis container via the logical machine axis image, then **all** channels concerned see **different axes** after a rotation. This means: The time for a rotation must be coordinated between the channels. This is performed by means of the available language commands.

---

Each entry in the axis container must be assigned to the correct channel at all times. The system variables in 2.6.1 offer the possibility for the parts program or synchronized action to gain information about the current axis container state.

### Commands for the axis container rotation

The requirement outlined above for coordinating channels that jointly use an axis container is contained in the effects of the command **AXCTSWE**.

Notation:

| | |
|---|---|
| **AXCTSWE**(CTi) | ;The function name stands for:<br>;AXis ConTainer SWitch Enable |

CTi is the identifier of the axis container which must be advanced. The increment must be stored in setting data

SD 41700: AXCT_SWWIDTH[container number]

(container-specific). SD 41700: AXCT_SWWIDTH (AXis ConTainer SWitch WIDTH) is available to all NCUs via the link module (i.e. all NCUs connected via a link module see the same values).

Function:
Each channel whose axes are entered in the specified container issues an **enable for a container rotation** if it has finished machining the position/station. If the enables for **all** channels for the axes of the container have been received, container rotation takes place with the increments set in SD 41700: AXCT_SWWIDTH[container number] (the direction of rotation is also assessed if there is a leading sign).

The following variant is provided to simplify start-up:

| | |
|---|---|
| **AXCTSWED**(CT1) | ;The function name stands for:<br>;AXis ConTainer SWitch Enable Direct |

The axis container rotates according to the settings in setting data SD 41700: AXCT_SWWIDTH[container number]. This call may only be used if the other channels, which have axes in the container are in the **RESET** state.

**Note**

In **SW 5.2** and higher, the axis container names assigned in
MD 12750: $MN_AXCT_NAME_TAB
can be used for commands AXCTSWE and AXCTSWED.

In the earlier software version **SW 5.1**, **channel axis identifiers** must be specified which refer to the container to be rotated via the logical machine axis image. The rotation must be specified in a separate command AXCTSWE(channel axis) for each container.

**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) for an appropriate container axis
- The next AXCTSWE(CTi) 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 increment even though absolute dimension is 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**

SD 41700: AXCT_SWWIDTH[container number] is only updated for new configurations. If after the incremental rotations of the RVM/MS the position has reached a switching position before the original position, the container can continue to be rotated **forwards**, in order to reach the original 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 interrupted.

**Home channel of a container axis**

If more than one channel has access authorization (i.e. a “reference”) to the axis due to the setting in MD 20070: AXCONF_MACHAX_USED, the write access to the axis (setpoint input) can be passed on. Machine data MD 30550: AXCONF_ASSIGN_MASTER_CHAN creates a default assignment between an axis and a channel. MD 10002: AXCONF_LOGIC_MACHAX_TAB is set to define which NCU possesses the axis after power-up or is producing the interpolation value. Since the axial machine data for link axes are identical on all NCUs, MD 30550: AXCONF_ASSIGN_MASTER_CHAN is evaluated only if the NCU has write authorization to the axis (see logical machine axis view in MD 10002: AXCONF_LOGIC_MACHAX_TAB).

**Axis replacement**

Passing the write authorization to an axis (setpoint input) by means of Get, Release,..., works for a container axis in the same way as for a normal axis. Write authorization can only be replaced between the channels of **one** NCU. Write authorization cannot be passed beyond the boundaries of an NCU.

## 2.6.1 System variables for axis containers

**States of an axis container**

The following system variables allow parts programs and synchronized actions to access information about the current state of an axis container.

**Legend:**

| | |
|---|---|
| r | Read |
| TP | Part program |
| SA | Synchronized action |
| SW | Software version |
| n | Axis container identifier with SW 5.2 and channel axis identifier with SW 5.1 |

| Name | Type /SW | Description/values | Index | PP access | SA access |
|---|---|---|---|---|---|
| **$AC_AXCTSWA[n]** (AXis ConTainer SWitch Active) | BOOLEAN /5 | **Channel** status of axis container rotation/<br>1: The channel has enabled axis container rotation for axis container n and this rotation is not yet complete.<br>0: The axis container rotation has ended<br>Examples: See Chapter 6 “Axis container coordination” | Names | r | r |
| **$AN_AXCTSWA[n]** | BOOLEAN /5 | **Axis container rotation**/<br>1: An axis container rotation is executed immediately by axis container n<br>0: No axis container rotation is active<br>Examples: See Chapter 6 “Axis container coordination” | Names | r | r |
| **$AN_AXCTAS[n]** (AXis ConTainer Actual State) | INT /5 | **Current rotation of axis container**<br>The number of slots by which the axis container has just been rotated is specified for axis container n./<br>0 to (max. number of occupied slots in axis container –1)<br>The default setting is entered after power ON. This is the value 0. | Names | r | r |

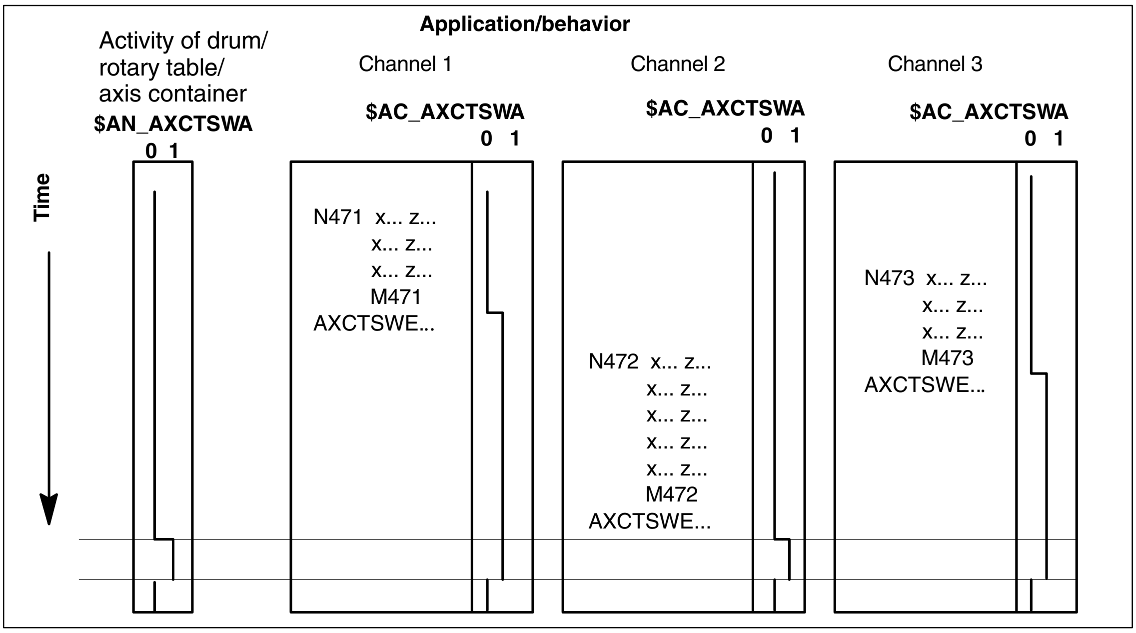



Fig. 2-13 Axis container rotation dependent on enable by channels concerned

### 2.6.2 Machining with axis container (schematic)

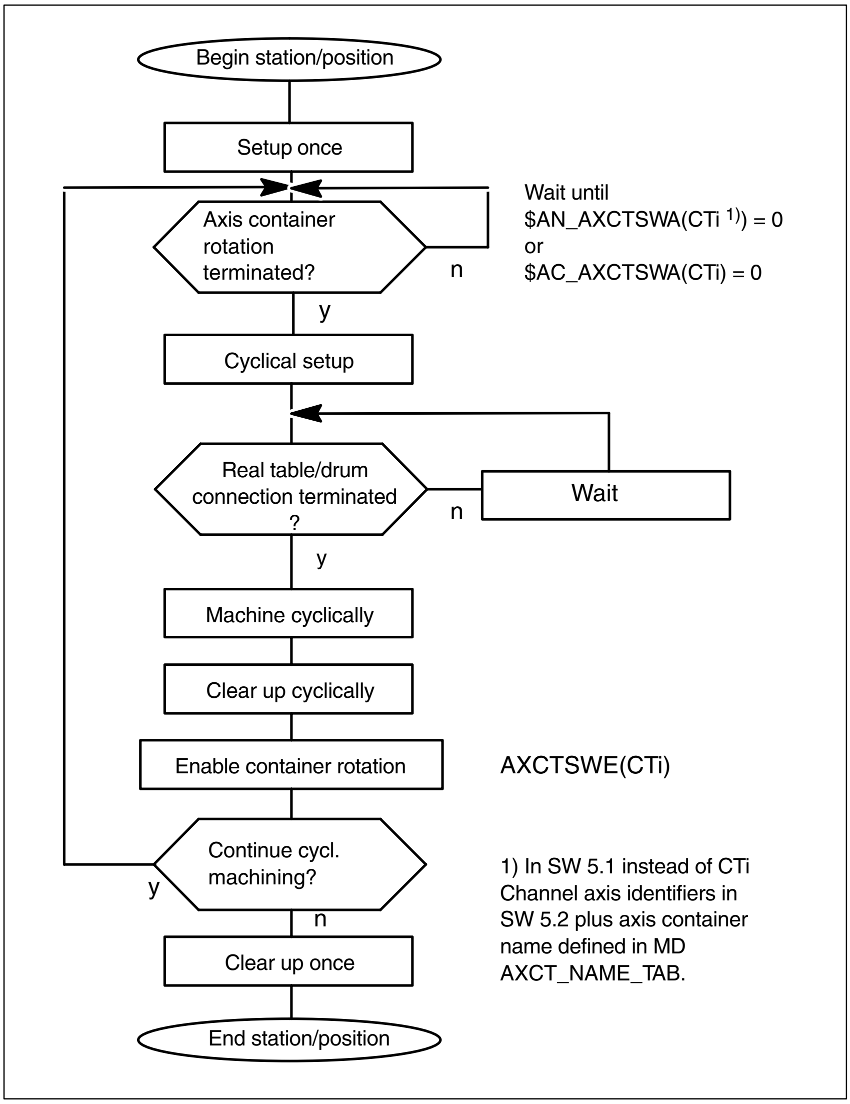



Fig. 2-14 Schematic machining of a station/position

Note:
An NCU machining cycle which is in charge of the rotation of the rotary table or the drum for multi-spindle machines contains the query of enables for container rotation of all NCUs concerned. If all enables are present, switching to the next position/station takes place. The axis containers are rotated accordingly.

### 2.6.3 Axis container behavior after power ON

The container always assumes the state defined in the machine data when the power is switched on, irrespective of its status as the power supply was switched off, i.e. the user must distinguish between the actual status of the machine and the default setting and compensate accordingly by specifying appropriate axis container rotations. He can do this, for example, by means of an ASUB containing AXCTSWED in one channel while the other channels are still in the RESET state.

### 2.6.4 Axis container response to mode switchover

A container axis in an axis container which has been enabled for rotation cannot be traversed in JOG mode. An axis container can only be rotated in JOG mode by means of an ASUB.

### 2.6.5 Axis container behavior in relation to ASUBs

An enabling command for axis container rotation cannot be canceled, i.e. if an axis container rotation has been enabled in an ASUB, the enabling command remains effective even when the ASUB has ended.

### 2.6.6 Axis container response to RESET

A reset cancels the enabling command for axis container rotation. The reset channel is then no longer involved in the axis container rotation. The enabling commands in the other active channels can effect a rotation. If all channels except one have been reset, the one remaining active channel can set the rotary position directly with AXCTSWED.

### 2.6.7 Axis container response to block searches

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.

### 2.6.8 Supplementary conditions for axis container rotations

**Note**

Through appropriate programming measures, the user must ensure that
– the right zero offsets are effective after the container switch and
– that no transformations are active during the container switch.

**Axial machine data**

If an axis is assigned to an axis container, then certain axial machine data must be identical for all axes in the axis container as the data are activated. This can be ensured by making a change to this type of machine data effective all container axes and all NCUs which see the axis concerned. The message: “Caution: This MD will be set for all container axes” is output at the same time.

During power-up, all axial machine data of this type are synchronized with the values of the machine axis in **slot 1 of the axis container**. In other words, the relevant machine data are transferred from the machine axis in slot 1 of the axis container to all other container axes. If machine data with other values are overwritten by this process, the message: “The axial MD of the axes in axis container <n> have been adapted” is output.

If a slot in the axis container is re-assigned (through writing of machine data MD12701–12716: AXCT_AXCONF_ASSIGN_TAB<n>), then the following message is output: “The MD of the axes in axis container <n> will be adapted on next power-up”.

Axial machine data of the type discussed above are identified by attribute ***containerEqual*** (equal for all axes in the axis container). With an NCU link, the axis container is defined **on the master NCU** (see Section 2.4).

**Axis states**

If a container axis is active in axis mode or as a positioning spindle (POSA, SPOSA) and its axis container needs to be rotated, then the rotation cannot be executed until the container axis has reached its end position.

A container axis which is active as a **spindle continues to turn** as the axis container rotates.

SPCON (switchover to position control) is attached to the physical spindle, i.e. this status is passed on with the spindle when an axis container rotates. SETMS (master spindle), on the other hand, refers to the channel and remains active in the channel when an axis container rotates.

**Continuous path mode G64**

An axis container rotation interrupts G64 mode in a channel in which a container axis in the rotating container is also a channel axis, even if it does not belong to the path grouping. This interruption does not occur, however, until an axis in the rotated axis container is programmed again.

**PLC axes**

If a container axis in a container which is enabled for rotation must become a PLC axis, then this status change request is stored, and the changeover to PLC axis status does not take place until **after** completion of the axis container **rotation**.

**Command axes**

A container axis in a container enabled for rotation cannot be declared a command axis. The traverse request is stored in the channel and executed on completion of the axis container rotation.

Exceptions to this rule are synchronized actions M3, M4, M5 and a motion-changing S function: If an axis container rotation is active and the spindle is transferred to the control of another NCU, alarm 20142 (channel %1 command axis %2: Invalid axis type) is output. These synchronized actions do not change a channel axis into a command axis, but leave it in its original state. Synchronized actions of this type cannot be stored.

**References:** /FBSY/ Description of Functions Synchronized Actions

**Reciprocating axes**

A container axis in a container enabled for rotation cannot become a reciprocating axis, i.e. this change in status does not take place until the axis container has finished rotating. The status change command remains active.

**Coupled axes**

An axis container cannot rotate while an axis coupling, in which one of its container axes is involved, is still active. The coupling must be deselected (COUPOF) prior to rotation and selected again (COUPON) afterwards. A new COUPDEF command is not necessary.

**Compile cycles**

In SW 5.2, a compile cycle axis cannot be a container axis.

**Main run offset values**

The main run offset values (DRF offset, online tool offset, synchronized action offset, compile cycle offset) for a channel axis assigned to a container slot remain valid after the relevant axis container has rotated. External zero offsets cannot remain valid after an axis container rotation as these refer to specific machine axes. If an external zero offset is active, the axis container rotation is rejected with alarm 4022.

**Axial frame**

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 thus also changes the axial frame**. If the two frames do not coincide, a synchronization process (internal REORG) is performed.

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 himself is responsible for ensuring that the correct frames are selected after a rotation by, for example, programming basic frame masks.

**Transformations**

If the container axis is a spindle which is involved in a transformation, then the **transformation** must be **deselected** before the axis container rotation is enabled. Otherwise alarm 17605 is activated.

**Gantry grouping**

Gantry axes cannot be axes in an axis container.

**Drive alarms**

When a drive alarm is active for a container axis, then the associated axis container cannot rotate until the alarm cause has been eliminated.

# 2.7 Cross-NCU user communication, link variables

**Introduction**

For large machine tools, rotary indexing machines and multi-spindle machines, whose movement sequences are controlled by more than one NCU, the applications on a single NCU must be able to exchange information rapidly with the other NCUs connected via link module.

For this purpose there are:

- Link variables

## 2.7.1 Link variables

**Definition**

Link variables are **system-global data** that can be addressed by the connected NCUs as **system variables** if link communication is configured. The
– **contents** of these variables,
– their **data type**,
– their **use**,
– their position (**access index**) in the link memory
are defined by the user (in this case this is usually the machine manufacturer).

**Prerequisite**

- To active NCU link communication, MD 18780: MM_NCU_LINK_MASK must be set.
- The link grouping must be installed and configured according to 2.8.

**Application**

As link variables are formally system variables, they can be read/write accessed in
**part programs** and in
**synchronized actions**
(as a rule).
Access possibilities for the individual link variables are specified under 2.7.2.

---

**Note**

On installations without an NCU link, the link variables can also be used NCU-locally as an additional means of cross-channel communication.

---

**Structure**

Each NCU connected to an entire system with link module sees a **link memory** in which the link variables are stored uniformly. Data exchange takes place after changes in the following interpolation cycle.

**Size of link memory**

The size of the link memory can be configured within the limits set by machine data

MD 18700: MM_SIZEOF_LINKVAR_DATA

It is necessary to define the same size for all connected NCUs. If there are deviations, the system adapts the link memory size of all NCUs according to the largest size specified. If the memory area of the link memory is exceeded during an access attempt, alarm 17020 is output.

**Initialization of link memory**

After power-up, the link memory is initialized with 0.

**Data types of link variables**

The link memory can contain link variables with the following data types:

- INT $A_DL**B**[i] ; Data **B**yte (8 bits)
- INT $A_DL**W**[i] ; Data **W**ord (16 bits)
- INT $A_DL**D**[i] ; Data **D**ouble word (32 bits)
- REAL $A_DL**R**[i] ; **R**eal data (64 bits)

According to the data type, 1, 2, 4, 8 bytes are addressed when reading/writing the link variables.

The position offset in bytes in the data area for global data is determined directly by the programmed field index. This is thus independent of the data type and specifies the offset in bytes.

**Ranges of values**

The data types have the following value ranges:

| | |
|---|---|
| BYTE: | –128 to 255 |
| WORD: | –32768 to 65535 |
| DWORD: | –2147483648 to 2147483647 |
| REAL: | –4.19e–308 to 4.19e–307 |

Alarm 17080 is generated when the upper value range limit is violated and alarm 17090 with violation of the lower value range limit and alarm 14096 in the case of an illegal type conversion.

The value range of these variables (with a negative value) applies only to write operations. Only the corresponding positive (unsigned) value can be read back.

**Addressing with access to global variables**

Index i always represents the distance in bytes from the beginning of the link memory. The index is counted from 0. This means that:

| Type | **Interpretation of (i)** (counting starts at 0 each time) |
|---|---|
| $A_DLB[i] | Byte i is followed by a data of type Byte. $A_DLB[7] addresses byte 8 from the beginning of the link memory. |
| $A_DLW[i] | Byte i is followed by a data of the word type. $A_DLW[4] addresses the word which is located on byte 5 from the beginning of the link memory. |
| $A_DLD[i] | Byte i is followed by a data of the double word type. $A_DLD[12] addresses the double word which is located on byte 13 from the beginning of the link memory. |
| $A_DLR[i] | Byte i is followed by a data of the real type. $A_DLR[24] addresses the real value which is located on byte 25 from the beginning of the link memory. |

**Link memory use**

The link memory can have different assignments for processes that are completely separated in time. The various NCU applications that access the link memory jointly **at any one time** must use the link memory **in a uniform way**.

**Access from synchronized actions**

If an impermissible index is used for access to the link memory from a synchronized action or a parts program, alarm 20149 is issued.

**Write access to link variables**

When writing to link variables of the link memory, for example as follows

$A_DLB[5] = 21,

a write element is required. The write element serves for communication with further NCUs which must see the modified contents in the link memory. Each write process to a link variable requires a write element. It is busy with the write process until the main run executing data exchange with the other NCUs is completed.

Since global data can be written by all channels and NCUs, the user must ensure proper coordination of write and read access operations. Variables are written immediately if an NCU link connection is active. Writing and immediate readback of a variable produces the same result. Variables are written only in synchronism with the main run. Writing and immediate readback in the same parts program block produces a different result.

**Number of write elements**

The write elements available for writing to link variables are limited. Their number is determined by:

MD 28160: MM_NUM_LINKVAR_ELEMENTS

If no more write elements are available for an intended write process, alarm 14763 is issued. The set number of write elements only limits the number of write processes that can be written in one block.

**Dynamic response during write**

Writing the link variables is immediately completed for the **local** NCU in the current interpolation cycle (in the sequence of commands). If the user does not exceed the number of possible write processes (can be checked in system variable $A_LINK_TRANS_RATE) in the current interpolation cycle, **all other** NCUs will have access to the written information 2 interpolation cycles later. If link variables are used exclusively to coordinate the channels of a multi-channel NCU, they can be written in the same interpolation cycle.

---

**Note**

The user (machine manufacturer) must ensure that the time is consistent for larger data blocks that are logically associated with one another. The transmission is word by word. The data quantity which can be transferred in the same interpolation cycle is specified in system variable $A_LINK_TRANS_RATE (see below). A transmission can be protected by marking variables of the link memory as semaphores.

---

## 2.7.2 System variables of the link memory

The following system variables are available for accessing the link memory:

**Legend:**

| | |
|---|---|
| r | Read |
| w | Write |
| R | Read with implicit preprocessing stop |
| W | Write with implicit preprocessing stop |
| TP | Part program |
| SA | Synchronized action |
| SW | Software version |

| Name | Type /SW | Description/values | Index | PP ac-cess | SA ac-cess |
|---|---|---|---|---|---|
| **$A_DLB[i]** | INT/5 | Addresses a byte in the link memory / 0 to $MN_MM_SIZEOF_LINKVAR_DATA –1 | Counters | R/w | r/w |
| **$A_DLW[i]** | INT/5 | Addresses a data word in the link memory / 0 to $MN_MM_SIZEOF_LINKVAR_DATA–2 | Counters | R/w | r/w |
| **$A_DLD[i]** | INT/5 | Addresses a data double word in the link memory / 0 to $MN_MM_SIZEOF_LINKVAR_DATA–4 | Counters | R/w | r/w |
| **$A_DLR[i]** | REAL/5 | Addresses a REAL value in the link memory / 0 to $MN_MM_SIZEOF_LINKVAR_DATA–8 | Counters | R/w | r/w |
| **$A_LINK_TRANS_RATE** | INT/5 | For synchronized actions: Number of bytes that can still be transferred in the current interpolation cycle via link communication. / –2147483648 to 2147483647 | | | r |

---

**Note**

Use of index i is described in detail in 2.7.1.

---

## 2.7.3 Link axis drive information

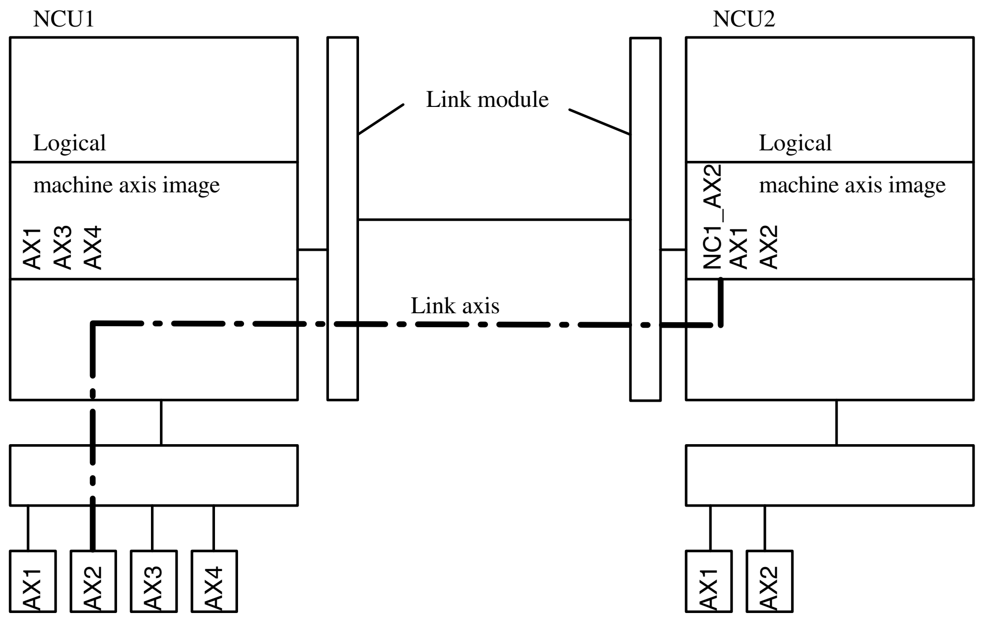
You can access the drive data via the machine axis identifier, even if the axis is being applied to another NCU. Software Version 6 and higher supports the use of the following drive system variables with machine axis identifiers [n] (only channel axis identifiers up to now):

– $AA_LOAD[n], $VA_LOAD[n]
– $AA_TORQUE[n], VA_TORQUE[n]
– $AA_POWER[n], $VA_POWER[n]
– $AA_CURR[n], $VA_CURR[n]
– $VA_VALVELIFT[n]
– $VA_PRESSURE_A[n]
– $VA_PRESSURE_B[n]

The following section describes how to query the drive status of link axes by means of these system variables and static synchronized actions.

**References:** /FBSY/

The connections are described by means of an example that is easily applicable to the requirements of the control at hand:




The machine axis identifier of the home NCU addresses the system variables of the link axis' drive information and passes them on via link variables.

Fig. 2-15 Passing on link axis drive information

**Prerequisite**

Machine data MD 36730: DRIVE_SIGNAL_TRACKING must be set to value 1.

**Sequence**

The drive system variable values of a link axis are provided in two steps:

1. The home NCU (that is, the NCU that is physically connected to the link axis) uses a static synchronized action to cyclically read the information contained in the system variables into a link variable (see 2.7.1). The link variable can also be accessed on the interpolating NCU (in our example, NCU2).
2. The interpolating NCU checks (e.g. in another synchronized action) the state of the link variable and initiates the required response.

**Machine data for example**

```
NCU1:
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = “AX1”
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = “AX3” ;*
; * If you move the axis that is not defined as a link axis
; closer towards the other axis, this would be AX2.
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = “AX4”
$MA_DRIVE_SIGNAL_TRACKING[AX2] = 1          ; Enabling
                                           ; initialization

NCU2:
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = “NC1_AX2” ; Link axis
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = “AX1”
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = “AX2”
```

**Synchronized actions**

```
NCU1:
; Static synchronized action cyclically transfers the drive variable
N111 IDS=1 WHENEVER TRUE DO $A_DLR[0]=$VA_CURR[AX2]

NCU2:
; Static synchronized action accepts value from link variable and
; triggers alarm 61000 if the current exceeds limit of 23A
N222 IDS=1 WHEN $A_DLR[0] > 23.0 DO SETAL(61000)
```

## 2.8 Configuration of a link grouping

**Introduction**

The preceding chapters described how to configure link axes and axis containers. Both require a link communication to be established between the NCUs concerned. Setting up the link communication takes place by means of:

- The link module hardware
  **References:** /PHD/, Configuring Guide NCU 571– 573.2
- Machine data

The following section describes how to use the required machine data.

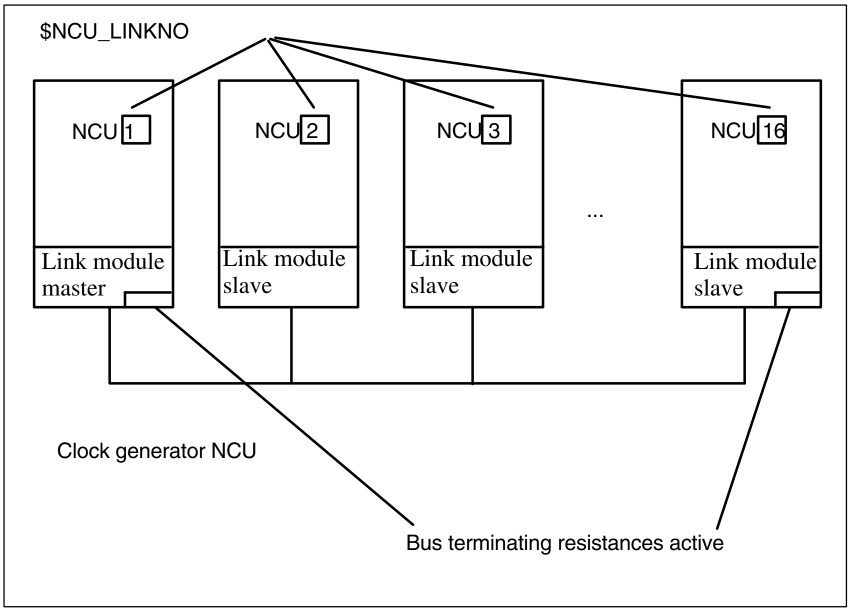



Fig. 2-16 Link grouping

**Link grouping**

A link grouping consists of a minimum of 2 and maximum of 16 NCUs interconnected by link modules.

- The link module master (MD 12510: NCU_LINKNO = 1) plays a leading role in this process. It synchronizes the interpolation cycle and sets up slave communication in power-up. It is advisable to assign the NCU numbers in continuous ascending order for the slave modules.
- The first and last module in the physical chain must activate the bus terminating resistances.
- The software version must be identical on all NCUs in a link grouping.

**Machine data**

Machine data
MD 18780: MM_NCU_LINK_MASK
ensures that link communication is established. It provides the dynamic memory space that is required for communication in the NCUs equipped with link modules.

Machine data
MD 12540: LINK_BAUDRATE_SWITCH
specifies the data transfer rate of the link communication with the following assignment:

| Set value | Rate | | |
|---|---|---|---|
| 0 | 9.600 | Kbaud | |
| 1 | 19.200 | Kbaud | |
| 2 | 45.450 | Kbaud | |
| 3 | 93.750 | Kbaud | |
| 4 | 187.500 | Kbaud | |
| 5 | 500.000 | Kbaud | |
| 6 | 1.500 | Mbaud | |
| 7 | 3.000 | Mbaud | |
| 8 | 6.000 | Mbaud | |
| 9 | 12.000 | Mbaud | Default setting |

Machine data
MD 12550: LINK_RETRY_CTR
specifies the maximum number of times the link communication is repeated when an error occurred during frame transfer.

Machine data
MD 12530: LINK_NUM_OF_MODULS
specifies the number of link modules taking part in the link communication

Machine data
MD 12510: NCU_LINKNO
assigns a logical link number to an NCU; this number is used for link identification in conjunction with link axes and link communication. Identifications can be assigned independently of the physical sequence of the modules in the link string. The module with NCU_LINKNO = 1 is **master**.

**Warning**

Assignment of NCU_LINKNO must be unambiguous. An alarm is issued if there is an error.

Machine data
MD 12520: LINK_TERMINATION
specifies for the software which NCUs correspond to the bus terminating resistances. The set values refer to the entries defined with MD 12510: NCU_LINKNO. 0 corresponds to the first definition, 1 to the second, etc. from MD 12510: NCU_LINKNO.

**Note**

MD 12520: LINK_TERMINATION need only be set for the prototype hardware of the link module. It is only meaningful for the software.

The NCUs that are **physically** connected at the **beginning** and **end** of the bus must activate the terminating resistors. This measure is necessary for the link communication to work.

Machine data

MD 30554: AXCONF_ASSIGN_MASTER_NCU

defines for the purposes of power-up which NCU in an NCU grouping will be responsible for generating the axis setpoint (master NCU).

Machine data

MD 30560: IS_LOCAL_LINK_AXIS

specifies that the axis drive needs to be started during power-up, even if the axis is operating under the control of another NCU. It is evaluated only if machine data required to create a link grouping have been set, but link communication has failed due to an error.

---

**Note**

It may be necessary to increase the interpolation cycle due to the number of link axes and write elements.

---

## 2.9 Communication in link grouping

Although communication by means of link modules is high-speed communication, the following aspects have to be taken into account during configuration.

**Data transport**

Both cyclic and acyclic services are used for data communication. The cyclical data area ensures that the data are transferred in every interpolation cycle. Other data are transferred in the acyclic service, e.g. machine data, link variables and data for container rotation. If too many data were selected simultaneously for the acyclic link communication, the transfer sequence is governed by an internally defined priority system. The actual value/setpoint data for the lead link axis are transferred via the cyclic service.

**Dependencies**

The times for communication are determined by:

– Number of data for cyclic exchange
(number of link axes, number of lead link axes (see further below), number of axes in a cross-NCU container with high priority and number of link variables having to be exchanged per each interpolation cycle)

– Type and speed of the NCUs

– Link module speed (currently 12 Mbaud)

– Point-to-point protocol: The cyclic communication load increases sharply with each additional NCU, as the schematic 2-19 illustrates.

The set interpolation cycle must be identical for all lead link NCUs in the axis grouping.

**Link resources**

MD 18781: NCU_LINK_CONNECTIONS controls the assignment of send/receive buffers for link connections. As many as 32 such buffers are available for each NCU.

A buffer of this kind is reserved on a priority basis for the setpoint/actual value transfer of a link axis. Only the remaining ones can be used for acyclic data exchange (alarm, container switch, link variable transfers). The maximum number of 32 must not be exceeded.

If MD 18781: NCU_LINK_CONNECTIONS is set with 0, the software itself determines the requirement as 25 connections.

A value not equal to 0 explicitly defines the number of acyclic connections to other NCUs.

**Examples**

Let an axis container contain 12 slots. Three axes are local on NCU1, three axes in each case are link axes that are on NCU 2, NCU3, and NCU4. The MD is set with 0 as the value.

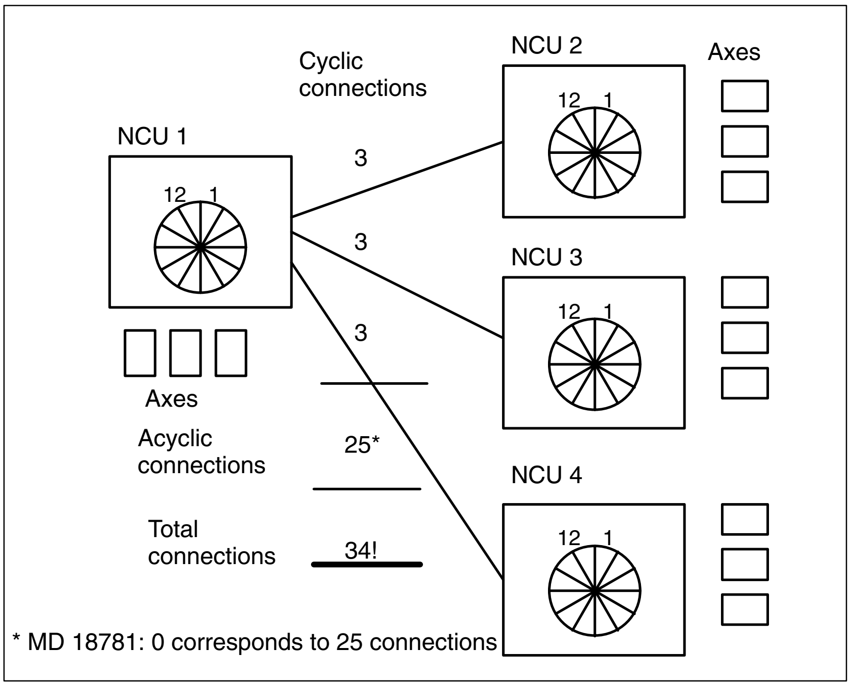



Fig. 2-17 Resources insufficient

Let an axis container contain 12 slots. Four axes are local on NCU1, four axes in each case are link axes that are on NCU 2, and NCU3. MD18781 is set with 9 as the value.

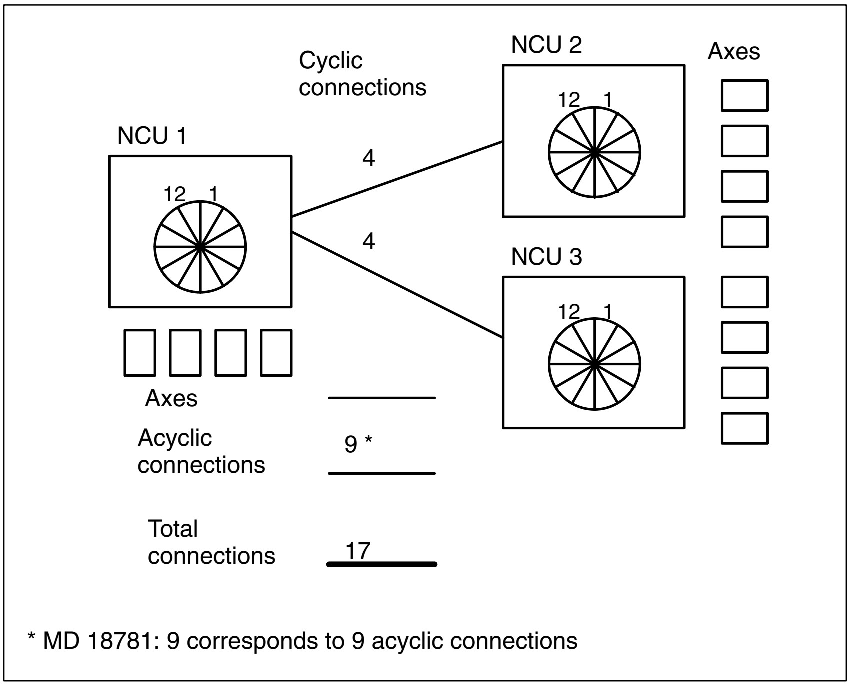



Fig. 2-18 Resources sufficient

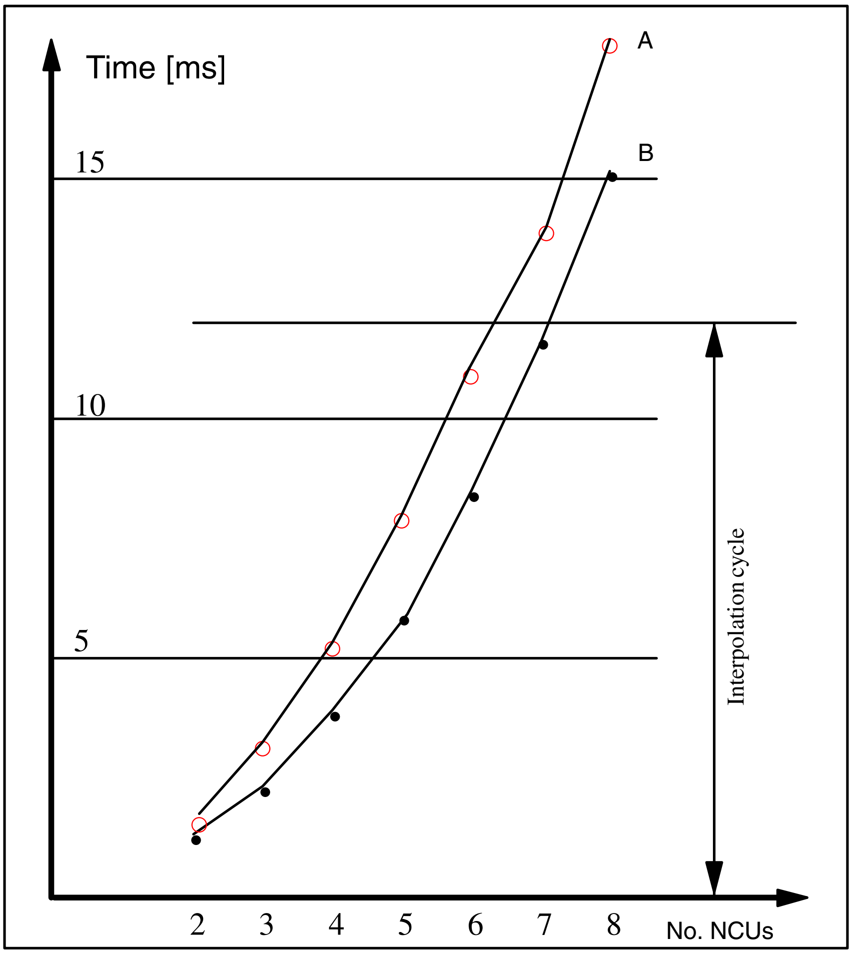



Fig. 2-19 Rise in Communication Time of the Number NCUs Connected over the Link (for scaling refer to Interdependencies)

**Configuration limit**

The figure above illustrates how the communication overhead grows as the number of NCUs increases.

Curve trace A:
Time required for the exchange of link variables/machine data information and the lead link axis information (one lead link axis) between the NCU giving the master value and other NCUs that interpolate the following axes as a function of the leading axis (lead link axis).

Curve trace B:
Time required for the exchange of link variables/machine data information between the NCU giving the master value and other NCUs that interpolate the following axes as a function of the leading axis (lead link axis).

**Rule:**
With a configuration, the time requirement must remain below the interpolation cycle according to curve trace A. If there is no longer any degree of freedom with respect to the number of necessary NCUs, the interpolation cycle might have to adapted.
If the interpolation cycle has to remain unchanged, the number of NCUs in the lead link axis grouping might have to be reduced.

## 2.10 Lead link axis

**Term**

A lead link axis allows read access to the axis data (setpoint, actual value, ...) on another NCU.

**Introduction**

The lead link axis concept offers a solution for the following problems:

The individual machining and handling stations are to move synchronous with or in relation to a common master value in so-called clocked sequences. The dependent axes are interpolated from another NCU, not the one interpolating the leading axis.

**Properties of a lead link axis**

Linking can be performed right across the NCU landscape,
i.e. several NCUs can be linked to a leading axis on another NCU.
The link system can encompass up to 8 NCUs interconnected via link communication. Axis values and other data are exchanged between the leading axis and the following axes via the link communication.
Typically, axes and spindles are on a par as the leading unit for coupled axes.

The same interpolation cycle must be set for all NCUs participating in the linkage.

**Restrictions**

- An axis (leading axis) that is referenced by a lead link axis cannot be a link axis, i.e. be moved by any other NCUs apart from its local NCU. See 2.5.
- An axis (leading axis) that is referenced as a lead link axis cannot be a container axis, i.e. addressed alternately by different NCUs. See 2.6.
- A lead link axis must not itself be a container axis.
- A leading link axis cannot be the programmed leading axis in a gantry grouping.
- Couplings with leading link axes cannot be cascaded.
- Axis replacement can only be implemented within the home NCU of the leading link axis. See below.

**Differential features from SW 5**

Lead link axes allow cross-NCU linkages where the master value axis and the following axes are programmed/interpolated on **different NCUs**. (See supplementary conditions).

Although it was possible to create a cross-NCU linkage in SW 5 by means of a link axis, programming and interpolating the leading value axis and following axes had to be performed on **one** NCU.

**Couplings**

The following linkage types can be used:

- Leading value (setpoint, actual value, simulated leading value)
- Coupled motion
- Tangential correction
- Electronic gear (ELG)
- Synchronous spindle

**Configuration leading axis NCU**

The lead link axis that is being interpolated as a leading axis on the NCU is configured on the interpolating NCU as a standard local axis. It is configured by means of channel axis and machine axis machine data, and via the logical machine axis image. In addition, there are the following requirements:

- The machine axis must be identified as lead link axis in MD 30554: AXCONF_ASSIGN_MASTER_NCU
- Number of link module must be specified in MD 12510: NCU_LINKNO
- The link functionality must be activated in MD 18780: MM_NCU_LINK_MASK = 1
- Number of link modules must be specified in MD 18782: MM_LINK_NUM_OF_MODULES
- Size of servo buffer specified in: MD 18720: MM_SERVO_FIFO_SIZE = **4**

**Configuration following axis NCU**

In addition to configuring the standard channel/machine axis machine data, you need to configure machine axis machine data on the NCU that is deriving following axis movements from the non-local leading axis (lead link axis):

- Number of link module must be specified in MD 12510: NCU_LINKNO
- The link functionality must be activated in MD 18780: MM_NCU_LINK_MASK = 1
- Number of link modules must be specified in MD 18782: MM_LINK_NUM_OF_MODULES
- If nec. MD 18402: MM_NUM_CURVE_SEGMENTS
- If nec. MD 18404: MM_NUM_OF_CURVE_POLYNOMS
- Size of servo buffer specified in: MD 18720: MM_SERVO_FIFO_SIZE = **2**
- The lead link axis must be configured in the logical machine axis image with MD : AXCONF_LOGIC_MACHAX_TAB[i] = “$NC_m$_$AX_n$”
  This allows a relation to be established with the NCU that is interpolating the lead link axis.
- The lead link axis must also be configured in the channel axes.

---

**Note**

For further information about link-specific machine data, please refer to the preceding subsections, Chapter 4, and the configuration and programming example provided in Chapter 6.

---

**Schematic sequence**

The figure below shows the constellation used in the example with the main data flows.
NCU 1 provides:
Interpolation and position control for master value axis or supplies the lead link axis of NCU2 with setpoint and actual value.
NCU 2 provides:
Coupling between lead link axis and master value axis (of NCU1) and generation of a dependent axis motion (following axis).

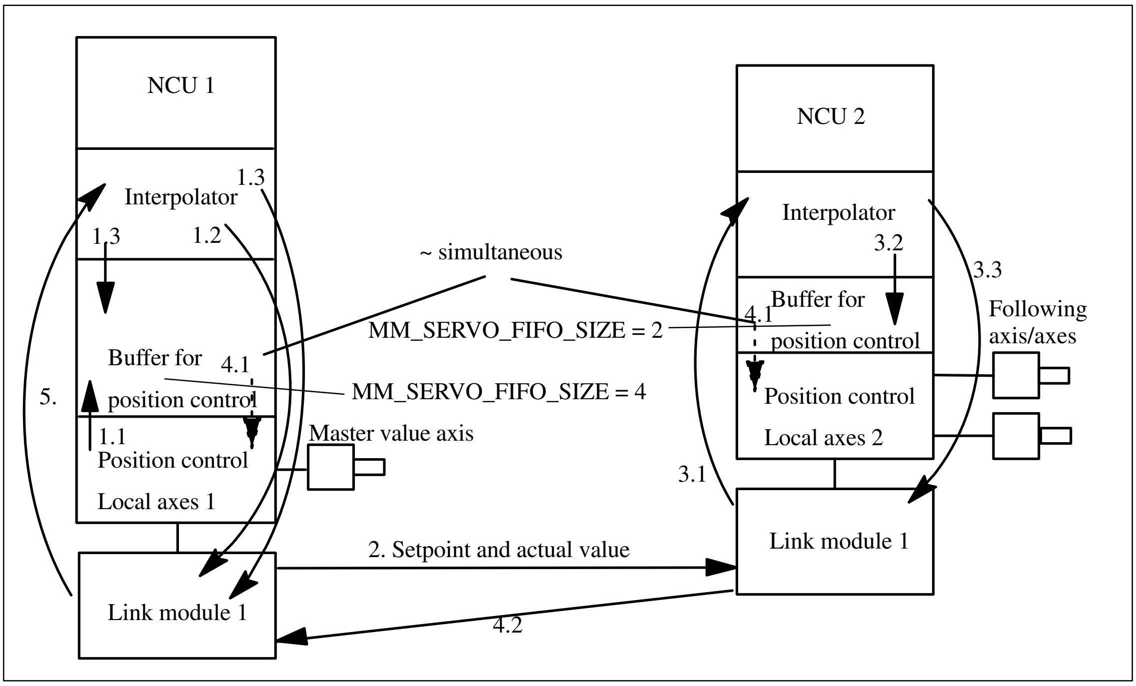



Fig. 2-20 Data flows for leading value axis, lead link axis and subordinate following axis/axes

The following steps are illustrated:

– 1.1 Position control on NCU1 reads in actual values of leading value axis from the drive and writes them in the communication buffer for interpolation.

– 1.2 In the same cycle as (1.1), the actual values from the NCU1 interpolator are written to the link module.

– 1.3 NCU1 interpolator writes the generated setpoints of the leading value axis to the link module and in the local buffer for position control.

– 2. Setpoints and actual values are transferred from NCU1 to NCU2 via the link module.

– 3.1 NCU2 interpolator receives setpoints of the leading value axis via the link module and calculates the setpoints of the following axis.

– 3.2 NCU2 interpolator writes the generated setpoints of the following axis in the buffer for position control.

– 3.3 NCU2 interpolator sends reaction feedback of following axis to leading value axis to the link module.

– 4.1 NCU1 position control receives setpoint for leading axis (delay caused by: MM_SERVO_FIFO_SIZE=4). NCU2 position control receives setpoint for the following axis.

– 4.2 Reaction feedback is transferred from NCU2 to NCU1.

– 5. NCU1 interpolator receives reaction feedback and can incorporate it when calculating setpoints for the leading value axis.

**Transporting lead link data**

The system variables $AA_LEAD_SP and $AA_LEAD_SV (see below) are transferred via the acyclic service. These system variables have a lower transfer priority than the link variables.

Approx. 320 bytes of data are exchanged for the lead link axis (setpoints and actual values). The communication time required for this operation is approx. equal to that required for a link axis.

In contrast to local NCU linkages, with cross-NCU linkages the leading axis setpoints (on NCU1) are delayed to achieve synchronously clocked assignments of setpoints for the leading/following axis. (See step 4.1.)

The set interpolation cycle must be identical for all lead link NCUs in the axis grouping.

**Axis replacement**

The channels of the NCU that is driving the following axes are not allowed to traverse or replace a lead link axis. The real leading axis can be replaced on its home NCU.

The commands GET, GETD and also Auto-Get ( $MA_AUTO_GET_TYPE) are rejected by the alarm “Channel %1 axis replacement for axis %1 not allowed”.

### 2.10.1 Programming a lead link axis

**Leading value axis view**

Only the NCU that is physically assigned to the leading value axis can program traversing movements for this axis. The travel program must not contain any special functions or operations. The configuration summarized in the preceding pages allows the control to perform the required setpoint delay automatically so that the leading axis and following axes are interpolated synchronously on the other NCUs.

**Following axes view**

The programming on the NCU of the following axes must not contain motion commands for the leading value axis (master value axis). Any violation of this rule triggers an alarm.

The leading value axis is addressed in the usual manner via the channel axis identifier. The states of the leading value axis can be accessed via special system variables.

**System variables**

The following system variables can be used in conjunction with the channel axis identifier of the leading value axis:
$AA_LEAD_SP ; Simulated leading value – position
SAA_LEAD_SV ; Simulated leading value – velocity

If these system variables are updated by the home NCU of the master axis, the new values are also transferred to any other NCUs which wish to control slave axes as a function of this master axis.

## 2.11 NCU link with different interpolation cycles

**Problem description**

In the real industrial world, parts deviating from a precise round/cylindrical shape are also required. (Example: Pistons that are oval in the manufacturing state. The operating temperature gives them their required almost round shape during use). The eccentric shapes are, e.g.:

- Oval shapes
- Trefoil shapes
- Cam
- More complex shapes

If you want to manufacture parts belonging to this category cost-effectively on turning machines, you can use the link functions described below, which are an expansion of the link functions described in 2.4 and the following sections. However, the boundary conditions must be complied with.

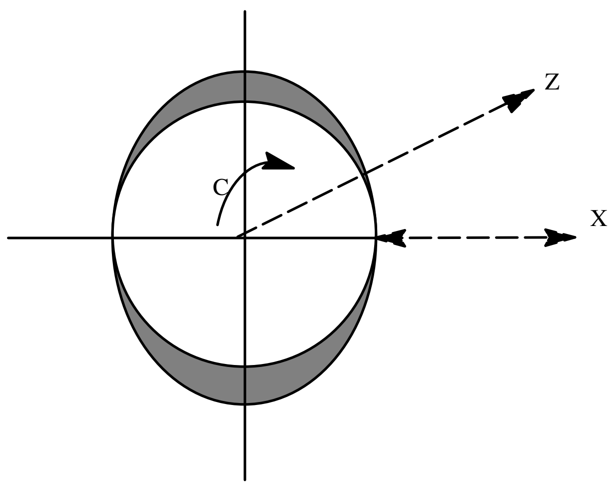



Fig. 2-21 Example of oval eccentric turning

**Sequence of motions**

While the workpiece is rotating about the C axis, the X axis must be advanced with high precision between the smallest and largest radius/diameter according to the required shape (sine, double sine, etc.). In general, a smaller linear movement per rotation is sufficient in Z direction. If the workpiece is to be moved at a typical speed (e.g. 3000 rpm), the X axis must be highly dynamic. This means that

– shorter interpolation cycle

– shorter position control cycle

compared to the requirements of the C and Z axis.

**Solution**

The requirements described in the motion sequences are met by a NCU link grouping; where the NCU with the shorter interpolation cycle drives the highly dynamic X axis and another NCU with standard interpolation cycle drives the less dynamic C and Z axes. General polynomials are used to specify the motion for the X axis.

---

**Note**

The NCU constellation with different interpolation cycles – where one is always faster than the others – is called “**FAST-IPO-LINK**” in alarm messages.

---

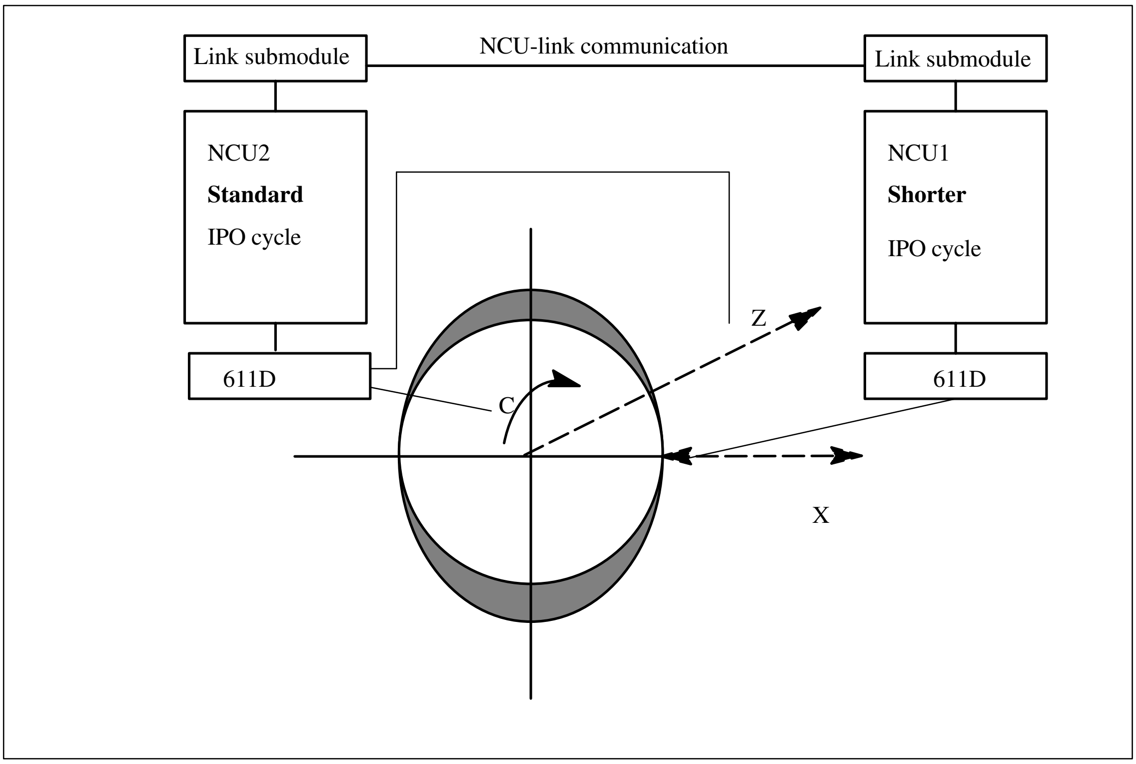



Fig. 2-22 NCU link with different interpolation cycles

**Generalized solution**

In a link grouping with several (up to 8) NCUs, some NCUs are set up with short interpolation cycles, some with standard interpolation cycles, and the axes are configured as in Fig. 2-22. The requirements for NCUs can be optimized in this manner. Eccentric machining is possible for multi-spindle turning machines in conjunction with the axis container concept (see 2.6).

---

**Note**

Alternatively, you can specify the motion for the X axis as follows:
Master value coupling of the X axis to the spindle (LEADON). This alternative is still possible and is sufficient for tasks with low dynamic performance requirements. However, it does not offer a specific function extension for eccentric turning, and does not allow setpoints to be specified on a NCU with faster IPO cycle via a NCU link.

**Reference:** /PGA/, Programming Guide Advanced

---

## 2.11.1 Diagram of general solution

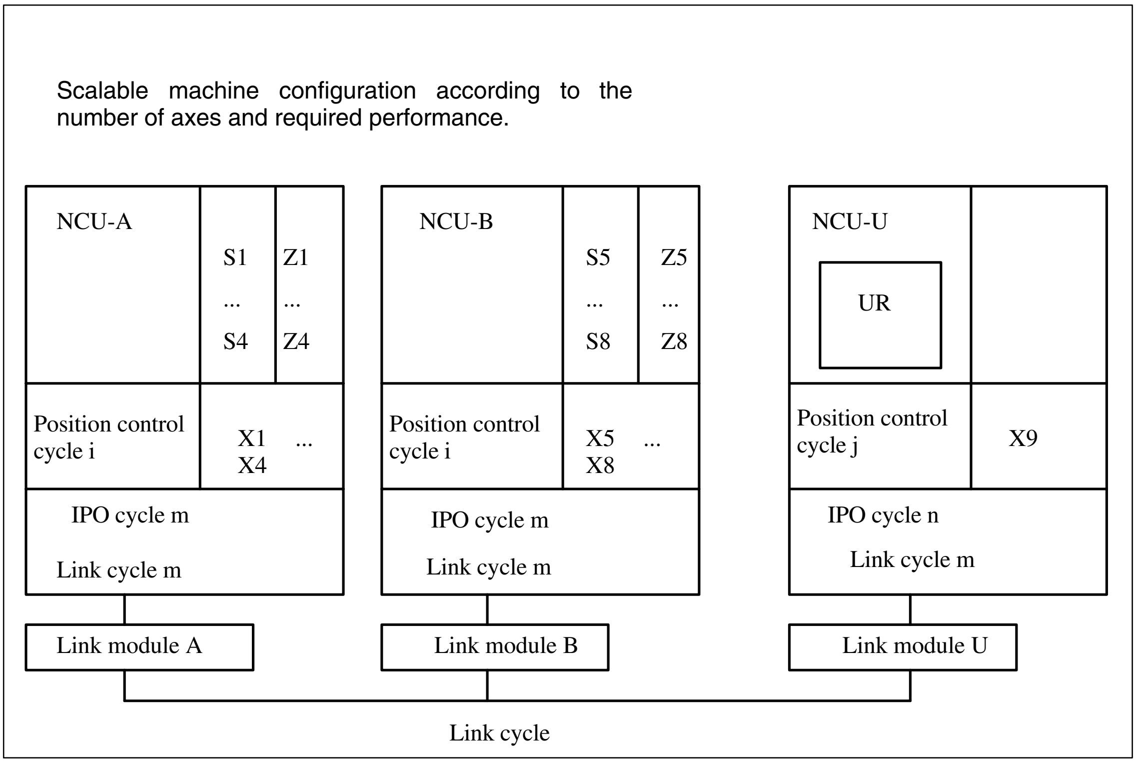



Fig. 2-23 Schematic example of a configuration with several NCUs and eccentric machining unit

**Abbreviations and terms**

| | |
|---|---|
| NCU-A, NCU-B | NCUs with standard interpolation cycle |
| NCU-U | Eccentric NCU with fast interpolation cycle |
| Position control cycle i | “slower” position control cycle |
| Position control cycle j | “faster” position control cycle |
| IPO cycle m | “slower” interpolation cycle |
| IPO cycle n | “faster” interpolation cycle |
| Link cycle m | Uniform link cycle from NCU-A to NCU-U |
| S1–S8 | Spindles 1–8 |
| Z1–Z8 | Z axes 1–8 |
| X1–X8 | X axes for concentric machining via NCU-A/NCU-B |
| X9 | X axis for eccentric machining |
| UR | On NCU-U channels are defined which drive the spindles and Z axes of NCU-A/NCU-B as link/container axes and interpolate the highly dynamic X axis with fast interpolation and position control cycles. |

**Cycles**

IPO cycle m is an integral multiple of IPO cycle n.
Position control cycle i may be greater than the fast IPO cycle n.
The slow IPO cycle m is used as communication cycle between NCU-A/NCU-B and NCU-U.

**Spindle speed**

The parts program manufacturer must first perform an OFFLINE calculation to ascertain the maximum spindle speed to program so that the axes (especially the X axis) are not overloaded when accelerating. If this maximum speed is incorrect, the spindle speed is reduced at parts where one of the participating axes would be overloaded. This in turn has a bearing on contour precision and should therefore be avoided.

**Contour deviations**

Spindle decelerations/accelerations lead to deviations from the programmed contour because the cycles of NCU-A/NCU-B and NCU-U have different lengths; therefore, the acceleration processes should already be terminated before initiating eccentric contour machining.

**Axis assignments**

Link axes may only be connected to NCUs that have the same IPO cycle as the link cycle.
Example: Axis X9 on the fast NCU-U in Fig. 2-23 cannot be interpolated as a link axis by NCU-A or NCU-B. However, axis Z4 can be interpolated by NCU-A, NCU-B and NCU-U.

**Setpoint delay**

From NCU-U's point of view, it is interpolating with the local X axis and the two link axes C and Z when performing eccentric machining. To achieve a correct contour, it is necessary to compensate for the time delay – which occurs when the setpoints are transferred to the link axes — and the differences in clock cycles by delaying the setpoints to the local X axis. This can be done by setting MD 18720: MM_SERVO_FIFO_SIZE.

Use the following formula for assigning MM_SERVO_FIFO_SIZE:

MM_SERVO_FIFO_SIZE = 2 * link cycle / IPO cycle +1

This formula is only applicable for NCU-U and NCU-A/NCU-B as in Fig. 2-23.

Examples:

NCU-U:
Link cycle = 16 ms

IPO cycle n = 2 msec

The cycle ratio is therefore 8; the value of the formula 2 * 8 + 1 = 17

NCU-A/NCU-B:

Link cycle = 16 msec

IPO cycle m = 16 msec

The cycle ratio is therefore 1; the value of the formula 2 * 1 + 1 = 3

In some cases, an alarm is generated to indicate setting errors in MD 18720: MM_SERVO_FIFO_SIZE.

**Effect of the SERVO_FIFO_SIZE settings**

– All functions that use actual values when interpolating for setpoint generation will be affected by the delay of the slower link cycle instead of the fast IPO cycle. This also applies to the response at faults (alarms that are issued to disable a mode group or for interpolatory braking).
– Alarm responses that are triggered directly in the servo are not affected by any additional delay.
– Acceleration processes on the link axes are always output with the link cycle and are therefore not exactly synchronous with the axes physically connected to the fast NCU. Therefore, it is only advisable to use this configuration with interpolating axes on NCUs with different IPO cycles if you are machining with link axes that are only accelerated a little or not at all. Thus the axes with high dynamic requirements must be physically connected to the fast NCU.
– If high precision is required for the machining operation, the tool should not have contact with the workpiece during acceleration/deceleration of the axes. Otherwise contour violations will result.

– It is only permissible to switch axis containers with **rotating** spindles on NCUs with IPO cycles that are equal to the link cycle. If it is possible to delay the container rotation, switching is delayed until the spindle is motionless. Otherwise alarm 4019 is issued. Axis container switching with **motionless** axes and spindles is also permissible on the faster NCUs. (See 2.6).

### 2.11.2 Different position control cycles

The general solution described in 2.11.1 also allows different position control cycles for NCU-A/NCU-B and NCU-U as in Fig. 2-23.

By setting the parameters as described below, you can improve the quality for some controller types.

When interpolating, the different cycle times are detected in fine interpolation and in the position control due to the different position control cycles; they are compensated for internally by delaying the setpoints for the axes with the faster position control cycle.

**Standard result**

If the axis with the slower position control cycle is not accelerated or is only accelerated very slowly, contour errors are largely avoided. However, contour errors will always occur when accelerating, therefore accelerations should not take place while machining.

**Special MD settings**

In some cases, it may be necessary to be able to adapt the delay (see examples below).
MD 10065: POSCTRL_DESVAL_DELAY has been added for this purpose. This allows you to adapt the setpoint delay in the position controller **for the entire NCU**. You can use positive and negative values. The set values (time in seconds, max. +/– 0.1sec) are added to the automatically calculated values. See under “Information”.

**Setting notes**

The required position setpoint delay depends on the controller structure used (DSC (dynamic stiffness control), feedforward control); this delay is taken into account when switching to the respective controller (e.g. FFWON (travel with feedforward control ON)). The **automatic** setting function obtains the maximum value for the position control cycles of the individual NCUs and delays the times on all the NCUs, thus providing compensation at constant speed for the different cycle times on all types of controller structures. In this manner, an additional delay can even be active on the axis with the slowest position control cycle.

The controller structure with

the most compensation is operation of torque feedforward control or with speed feedforward control and activated ramp encoder in the drive,

the least compensation is DSC or if feedforward control is deactivated.

**Information**

The setpoint delay for the three controller structures

- Without feedforward control (index 0)
- Speed feedforward control (index 1)
- Torque feedforward control (index 2)

is displayed in the read-only machine data MD 32990: POSCTRL_DESVAL_DELAY_INFO.

Negative values in MD 10065: POSCTRL_DESVAL_DELAY reduce the values for **all three** controller structures; positive values increase them.

**Appropriate changes to parameter settings**

It is appropriate to change the settings in MD 10065: POSCTRL_DESVAL_DELAY in the following cases:

- DSC is always activated or always run without feedforward control in axis mode.
  In these cases, additional delay is only necessary on the NCU with the fast position controller cycle. Then the following steps are appropriate:
  1.) MD 32990: POSCTRL_DESVAL_DELAY_INFO[0] on any axis on the NCU with the slowest position controller cycle.
  2.) Subtract this time from the effective value in MD 10065: POSCTRL_DESVAL_DELAY on <u>all</u> NCUs.
- With the typically used controller structure, there will be a delay in MD 32990: POSCTRL_DESVAL_DELAY_INFO in the axis with the fast position control cycle; the delay is not an integral multiple of the position control cycle.
  As there are particularly high dynamic performance requirements on the axis with the fast position control cycle, even the necessary averaging for generating the delay may lead to problems.
  Example: Typically, machining is conducted with speed feedforward control. In the axis X on NCU-U in Fig. 2-23 a value of 7 msec is read out in MD 32990: POSCTRL_DESVAL_DELAY_INFO[1]; the position control cycle on this NCU is 2 ms.
  In this case it may be practical to increase MD 10065: POSCTRL_DESVAL_DELAY by 1msec on <u>all</u> NCUs. The value of 8msec resulting in MD 32990: POSCTRL_DESVAL_DELAY_INFO[1] is an **integral** multiple of the position control cycle of 2msec.

### 2.11.3 Supplementary conditions

- The option “different interpolation cycle” can only be used in conjunction with NCU link (options, dependent on the axis number). The connected NCUs must all be fitted with the link module hardware components.
  - The configuration rules according to 2.8 apply.
  - Link axes must be configured as described in Subsection 2.5.1.
  - Axis containers can be used as described in Section 2.6.
- It is only permissible to use link axes on NCUs whose IPO cycle length is the same as that of the link cycle,
- Axis container
  - It is only permissible to switch axis containers with rotating spindles on NCUs with IPO cycles that are equal to the link cycle.
  - Axis containers can also be switched with motionless axes and spindle on the NCUs with faster IPO cycles.
- Acceleration and deceleration processes on axes with slower interpolation or position controller cycles will cause contour errors.

### 2.11.4 Activating NCU links with different interpolation cycles

- Setting different IPO cycle times in one link grouping
- Setting bit 1 in MD 18780: MM_NCU_LINK_MASK
- The option “NCU link with different interpolation cycles” is available.

**Example**

Please refer to Chapter 6 for an example of eccentric turning using the “NCU link with different interpolation cycles” function.

### 2.11.5 Different IPO cycles, behavior at power ON, RESET, etc.

The necessary setpoint delay on the servo causes a delay for two link cycles during boot, at reset, at New Config, etc. At reset, an additional delay of at least two link cycles is required to account for the necessary synchronization.

### 2.11.6 System variable with different interpolation cycles

There are no new system variables.

The existing general system variable $A_LINK_TRANS_RATE only displays a value not equal to zero in the **link communication cycle** on an NCU with an IPO cycle with a shorter length than the link communication cycle. The displayed value is available to the user; the transfer is only really conducted in this cycle.

If it is not possible to transfer all the link variables in one cycle, alarm 14764 "NCU link cannot transfer all link variables at once" is displayed as previously. This alarm can be suppressed.

## 2.12 Link grouping system of units

**Introduction**

Cross-NCU interpolations are possible in the link grouping with:

- Link axes (see 2.5)
- Lead link axes (see 2.10)
- Link axes on NCUs with different interpolation cycles (see 2.11)

With all constellations, it is only possible to achieve a correct result if all the NCUs connected via the NCU link are using the same system of units.

**Link grouping, global setting**

To ensure consistency across the configuration, if the **system of units** is switched over on the operator panel of one NCU in the grouping, with SW 5 and higher this affects the entire NCU grouping.

If the conditions:

- MD 10260: CONVERT_SCALING_SYSTEM=1.
- Bit 0 of MD 20110: RESET_MODE_MASK set in every channel
- All channels in the reset state
- Axes are not being traversed via JOG, DRF or the PLC
- Constant grinding wheel peripheral speed (GWPS) is not active

are met on all connected NCUs, the switchover is valid for all NCUs. If one of the switchover conditions for the system of units is not fulfilled on one or more NCUs, none of the NCUs will switch to the other unit system.

Actions such as parts program start or mode change are disabled for the duration of the switchover.

**Different systems of units**

It is only possible to use different unit systems in time intervals where there are no cross-NCU interpolations even though the link grouping is installed. The necessary settings must then be made NCU-specifically via G codes, as described in

**References:** /FB/, G2, Velocities, Setpoint/Actual Value Systems, Closed-Loop Control.

■

# Supplementary Conditions 3

## 3.1 Several operator panels and NCUs with control unit management option

**Configuring**

The number of configurable control units/NCUs is only limited by the availability of bus addresses on the individual bus segments of the different bus types.

The restrictions relating to the current SW version specified in the catalog or release notes apply.

**Availability**

The control unit management option is available in **SW 5.3** and higher.

**SW 6.3**

The control unit management option is expanded to provide a link of up to **nine MMC**s to a total of **nine NCU**s. The conditions for use of this option stipulated in previous chapters apply as before.

## 3.2 Several operator panel fronts and NCUs, standard functionality

**Availability with SW 3.1 and later**

**"Several operator panel fronts and several NCUs"configuration:**
Available in the basic version. The number of NCUs that can be connected is limited to 1 and the operator panel fronts to 2. One of the operator panel fronts must be an OP030.
**Programming language Step 7**
can be used.

**Expansions SW 3.2 or higher**

**Configuration "1 operator panel front and up to 3 NCUs"**
available according to Figs. 1-2 and 1-3.

When the link is created via the MPI line (187.5 kbps), a PLC-CPU315 must be used if the configuration includes more than one NCU since this PLC allows variable setting of the NC address.
Addresses must likewise be specified for exchange of data between PLCs via PROFIBUS-DP (PLC-CPU315 only) or for global data (duplicate address assignments).

**Expansions SW 3.5 or higher**

**Configuration "1 operator panel front and up to 4 NCUs and locally 1 MMC each"**
available according to Fig. 1-4.
Data exchange between NC and PLC is now also available with the PLC-CPU 314.

**Expansions SW 4 or higher**

**Operation of the m:n connection**
via the channel menu (see Subsection 2.1.11) which can be selected via the "Switchover channel" key.
Prerequisite for the channel menu is a configuration via the NETNAMES.INI file (see /IAD/, Installation and Start-up Guide 840D, section MMC).
The channel menu function is an option.

**Bus connection:**
The address space (previously 0, ..., 15) has been extended to (0, ..., 31).

---

**Note**

If an address > 15 is used, all components connected to the bus must be capable of processing addresses between 0 and 31.

---

## 3.3 Link axes

**Availability**

1. Precondition is that the NCUs are networked with link modules.
2. The **link axis** function is an option that is available from SW 5 for NCU 573.2 in variants for 12 axes and 31 axes; it is required for each link axis (max. 32). It is determined by number.
3. The **axis container** function is an option that is available from SW 5 for NCU 573 12 axes/31 axes; it is required for each container. If a link module is employed, this function can be used without the additional option.

**References:** /PHD/, Configuring Guide NCU 571-573.2

## 3.4 Axis container

**Availability**

**Axis container** is an option that is available for the NCU 573.2 with SW 5 and higher. In cases where axis containers are configured for link axes, the supplementary conditions for such containers as defined in Section 3.3 "Link axes" also apply.

## 3.5 Lead link axis

**Availability**

Software Version 6 and higher supports cross-NCU coupling with lead link axes. The functionality builds on the NCU link. Therefore, the NCU link options must be installed.

## 3.6 NCU link with different interpolation cycles

**Availability**

SW 6 and higher offers the NCU link with different interpolation cycles as an option.

All requirements that apply for link axes must be fulfilled.
(See 3.3).

If the non-local axes – seen from the point of view of the NCU with the fast interpolation cycle – are container axes (e.g. with multi-spindle turning machines) the corresponding option (see 3.4) must also be installed.

■

# Notes

# Data Descriptions (MD, SD) 4

## 4.1 Machine data for several operator panel fronts

The following machine data is provided for functions with SW 3.2 and higher:

| | | |
|---|---|---|
| **10134**<br>MD number | **MM_NUM_MMC_UNITS**<br>Resource units for MMC communications partners that are possible at the same time | |
| Default setting: 6<br>(FM-NC: 4, NCU571: 3, 810D: 3,<br>MMC 100–103: 2, OP030: 1) | Minimum input limit: 1 | Maximum input limit: 10 |
| Changes effective after POWER ON | Protection level: 2 / 2 | Unit: – |
| Data type: DWORD | Applies from SW: 3.2 | |
| Meaning: | Number of simultaneously possible MMC communication partners with which the NCU can exchange data.<br>The value influences how many communications jobs can be managed by the NC.<br>The higher the value, the more MMCs can be simultaneously connected to the NC.<br>Depending on the value entered in the machine data, DRAM is made available in the NCU for this function. Instructions regarding the modification of memory areas must be observed (see FB2/S7/).<br>The unit of MD 10134 is a “resource unit”.<br>A standard OP030 requires 1 resource unit and an MMC 100/103 2. OEM variants may require more or fewer resources.<br>• Setting the value lower (than would normally be required by the number of connected MMCs) does not necessarily cause problems. Occasionally operations may not work if several communication-intensive operator actions (e.g. loading program) are being conducted at the same time: Alarm 5000 is displayed. The operator action must be repeated.<br>• If the value is set higher, more dynamic memory than necessary will be used up. If the memory is required for other purposes, the value should be reduced accordingly. | |

# 4.2 Machine data for link communication

## 4.2.1 General machine data

| | | |
|---|---|---|
| **10002**<br>MD number | **AXCONF_LOGIC_MACHAX_TAB[n]**<br>List of machine axes available on an NCU<br>(logical NCU machine axis image) | |
| Default setting: AX1, AX2, ... | Minimum input limit: – | Maximum input limit: – |
| Changes effective after POWER ON | Protection level: 1 / 1 | Unit: – |
| Data type: STRING | Applies from SW: 5 | |
| Meaning: | MD AXCONF_LOGIC_MACHAX_TAB maps channel axes onto:<br>1. Local axes (default: AX1, AX2 ... AX31)<br>The entry $MN_AXCONF_LOGIC_MACHAX_TAB[n] = AX3 assigns local axis AX3 to axis index n. (For n = 3 the default setting AX3 is available. Therefore MD blocks for Software Version 4 and lower are compatible in SW 5).<br>2. Link axes (axes that are physically connected to another NCU)<br>The entry $MN_AXCONF_LOGIC_MACHAX_TAB[n] = NCj_AXi assigns axis AXi on NCU j to axis index n (link axis).<br>Limits:<br>n Machine axis address (of local NCU) 1 ... 31<br>j NCU number 1 ... 16<br>i Machine axis address (of local/remote NCU) 1 ... 31<br>3. Axis containers which in turn contain local or link axes.<br>The entry $MN_AXCONF_LOGIC_MACHAX_TAB[n] = CTr_SLs assigns container r and slot s to axis index n.<br>Limits:<br>n Machine axis address (of local NCU) 1 ... 31<br>r Container number 1 ... 16<br>s Slot number (location) in containers 1 ... 32<br>If several NCUs refer to the same machine axis in the NCU link grouping via this MD, it is necessary to define in axial machine data MD 30554: AXCONF_ASSIGN_MASTER_NCU which NCU is the master NCU, i.e. which NCU is to generate the setpoints for the position controller after boot. | |
| MD irrelevant for ...... | Systems without link modules | |
| Application example(s) | Initial.ini (extract) on NCU3:<br>$MN_AXCONF_LOGIC_MACHAX_TAB[4] = NC5_AX7<br>CHANDATA(2)<br>$MC_AXCONF_MACHAX_USED[1] = 5<br>$MC_AXCONF_CHANAX_NAME_TAB[1] = MyAx_Y<br>Part program block “G0 MyAx_Y = 100” running on NCU3/channel 2 traverses the 7th axis of NCU 5. | |
| Related to .... | AXCT_AXCONF_ASSIGN_TABi (create entries in containers i)<br>AXCONF_ASSIGN_MASTER_NCU | |

| | | |
|---|---|---|
| **10065**<br>MD number | **POSCTRL_DESVAL_DELAY**<br>Position setpoint.Delay | |
| Default setting: 0.0 | Minimum input limit: –0.1 | Maximum input limit: 0.1 |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: s |
| Data type: DOUBLE | Applies from SW: 6 | |
| Meaning: | Setpoint delay in position controller.<br><br>A setpoint delay can be parameterized in the position controller with this machine data. This is used for the NCU link if different position control cycles are to be parameterized on the NCUs and these axes are still supposed to interpolate with one another.<br>This MD is for optimizing the automatic setting function. | |
| MD irrelevant for ...... | Missing NCU link option | |
| Related to .... | MD 32990: POSCTTRL_DESVAL_DELAY_INFO | |

| | | | | |
|---|---|---|---|---|
| **10087** **(from SW 6: 18720)** MD number | **SERVO_FIFO_SIZE** **MM_SERVO_FIFO_SIZE** Size of data buffer between interpolator and position control | | | |
| Default setting: 2 | Minimum input limit: 2 | | Maximum input limit: 3 | |
| Changes effective after POWER ON | | Protection level: 3 / 2 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 5.2 | |
| Meaning: | The MD specifies the size of the data buffer between the interpolator and position control. If several NCUs for rotary indexing machines/multi-spindle turning machines are connected via the NCU link, the value for all connected NCUs must be set to 3. This setting compensates for the difference in transmission time between local setpoints and set points from another NCU via the NCU link. | | | |

| | | | | |
|---|---|---|---|---|
| **11398** MD number | **AXIS_VAR_SERVER_SENSITIVE** Response of the AXIS-VAR server to errors | | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | Protection level: 2/7 | | Unit: – |
| Data type: BYTE | | | Applies from SW: 5 | |
| Meaning: | If the server cannot supply any values for an axis (e.g. because the axis concerned is a link axis), then it returns a default value (generally 0).<br><br>For the purpose of testing, this machine data can be used to set the axis data server sensitively so that it returns an error message rather than default values.<br>0: Default value<br>1: Error message | | | |
| MD irrelevant for ...... | Systems without link modules | | | |
| Related to .... | MM_NCU_LINK_MASK | | | |

| | | | | |
|---|---|---|---|---|
| **12510** MD number | **NCU_LINKNO** NCU number in an NCU group | | | |
| Default setting: 1 | Minimum input limit: 1 | | Maximum input limit: 16 | |
| Changes effective after POWER ON | | Protection level: 1 / 1 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 5 | |
| Meaning: | Number or name for identifying an NCU within an NCU grouping.<br>In an NCU grouping (NCU cluster), the NCUs are interconnected via a link bus. | | | |
| MD irrelevant for ...... | Systems without link modules | | | |
| Application example(s) | See Chapter 2 “Configuration of link axes” | | | |
| Related to .... | MM_NCU_LINK_MASK | | | |

| | | | | |
|---|---|---|---|---|
| **12520** MD number | **LINK_TERMINATION** NCU numbers for which bus terminating resistors are active | | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: 15 | |
| Changes effective after POWER ON | | Protection level: 1 / 1 | | Unit: – |
| Data type: BYTE | | | Applies from SW: 5 | |
| Meaning: | LINK_TERMINATION defines for which NCUs the bus terminating resistors must be activated by the link module for the timing circuit. | | | |
| Related to .... | MM_NCU_LINK_MASK | | | |

| | | | |
|---|---|---|---|
| **12530**<br>MD number | **LINK_NUM_OF_MODULES**<br>Number of NCU link modules | | |
| Default setting: 2 | Minimum input limit: 2 | Maximum input limit: 16 | |
| Changes effective after POWER ON | Protection level: 1 / 1 | Unit: – | |
| Data type: DWORD | Applies from SW: 5 | | |
| Meaning: | The machine data specifies how many link modules are taking part in the link communication. | | |
| MD irrelevant for ...... | Systems without link modules | | |
| Related to .... | MM_NCU_LINK_MASK | | |

| | | | |
|---|---|---|---|
| **12540**<br>MD number | **LINK_BAUDRATE_SWITCH**<br>Link bus baud rate | | |
| Default setting: 9 | Minimum input limit: 0 | Maximum input limit: 9 | |
| Changes effective after POWER ON | Protection level: 1 / 1 | Unit: – | |
| Data type: DWORD | Applies from SW: 5 | | |
| Meaning: | The baud rate selected for link communication is defined by the values entered here :<br>**Set value** **Rate**<br>0 9.600 Kbaud<br>1 19.200 Kbaud<br>2 45.450 Kbaud<br>3 93.750 Kbaud<br>4 187.500 Kbaud<br>5 500.000 Kbaud<br>6 1.500 Mbaud<br>7 3.000 Mbaud<br>8 6.000 Mbaud<br>9 12.000 Mbaud | | |
| MD irrelevant for ... ... | Systems without link modules | | |
| Related to .... | MM_NCU_LINK_MASK | | |

| | | | |
|---|---|---|---|
| **12550**<br>MD number | **LINK_RETRY_CTR,**<br>Maximum number of message frame repeats in event of error | | |
| Default setting: 4 | Minimum input limit: 1 | Maximum input limit: 15 | |
| Changes effective after POWER ON | Protection level: 1 / 1 | Unit: – | |
| Data type: DWORD | Applies from SW: 5 | | |
| Meaning: | Maximum number of message frame repeats in event of error. If an error is detected during data transfer, the transmission can be repeated as often as specified in the MD.<br><br>Baud rate Recommended repetition value<br><br>187.500 Kbaud 1<br>500.000 Kbaud 1<br>1.500 Mbaud 1<br>3.000 Mbaud 2<br>6.000 Mbaud 3<br>12.000 Mbaud 4 | | |
| MD irrelevant for ...... | Systems without link modules | | |
| Related to .... | MM_NCU_LINK_MASK | | |

| | | | | |
|---|---|---|---|---|
| **12701 ... 12716**<br>MD number | **AXCT_AXCONF_ASSIGN_TAB1[s] ... AXCT_AXCONF_ASSIGN_TAB16[s]**<br>List of axes in axis container 1, ..., 16 | | | |
| Default setting: “ ” (empty string) | | Minimum input limit: – | | Maximum input limit: – |
| Changes effective after POWER ON | | Protection level: 7 / 2 | | Unit: – |
| Data type: STRING | | | Applies from SW: 5 | |
| Meaning: | Assignment of an axis container slot (slot s) to a machine axis or link axis. Up to 32 slots can be assigned to axes in an axis container.<br>Notation for entries:<br>NCm_AXn with NCU number m: 1...16 and machine axis address n: 1... 31<br>Example:<br>NC2_AX1 ; The axis is on NCU2 where it is the 1st machine axis.<br>AX5 ; Local axis 5 with only one NCU; the axis container mechanism is used<br>; only by more than one channel on an NCU.<br>The reference to an axis container slot of a channel is defined by machine data<br>MD $MC_AXCONF_MACHAX_USED and<br>MD $MN_AXCONF_AXCONF_LOGIC_MACHAX_TAB.<br>The axis that is actually assigned at a specific time depends on the container rotation state. All **channels** that access an axis container use the entries stored here. If channels from different NCUs access this container, pay attention to consistency across the NCUs!<br>Example:<br>CHANDATA(1)<br>$MC_AXCONF_MACHAX_USED[4]=9<br>$MN_AXCONF_LOGIC_MACHAX_TAB[8]=“CT1_SL1”<br>$MN_AXCT_AXCONF_ASSIGN_TAB1[0]=“NC1_AX1”<br>$MN_AXCT_AXCONF_ASSIGN_TAB1[1]=“NC2_AX1”<br>This machine data is distributed via the NCU link.<br>If $MA_AXCONF_ASSIGN_MASTER_CHAN is set for a container axis, then $MC_AXCONF_MACHAX_USED in the assigned channel must refer to the entry in $MN_AXCONF_LOGIC_MACHAC_TAB. This contains a reference to the axis container slot to which this axis is assigned in the default setting. A power-up alarm is otherwise generated. | | | |
| Related to .... | AXCONF_LOGIC_MACHAX_TAB, | | | |

| | | | | |
|---|---|---|---|---|
| **12750**<br>MD number | **AXCT_NAME_TAB**<br>List of axis container names | | | |
| Default setting: “CT1”, “CT2”, ... “CT16” | | Minimum input limit: – | | Maximum input limit: – |
| Changes effective after POWER ON | | Protection level: 1 / 1 | | Unit: – |
| Data type: STRING | | | Applies from SW: 5.2 | |
| Meaning: | A name can be assigned to each axis container. The axis container rotation commands AXCTSWE and AXCTSWED can be used in conjunction with these names. If no container names are defined, channel axis names must be used (SW 5.1) which address the desired axis container via the logical machine axis image. This also applies to the addressing of system variables for axis containers.<br>The default axis container names are CT1, CT2, ..., CT16 | | | |
| Application example(s) | The name of the 1st axis container is AXCT1. Definition in MD:<br>$MN_AXCT_NAME_TAB[0] = “AXCT1” | | | |

| 18700<br>MD number | MM_SIZEOF_LINKVAR_DATA.<br>Size of the NCU link variable memory | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 100 |
| Changes effective after POWER ON | Protection level: 7 / 2 | Unit: – |
| Data type: DWORD | Applies from SW: 5 | |
| Meaning: | Defines the number of bytes of link memory for NCU-global data. | |

| 18720<br>MD number | MM_SERVO_FIFO_SIZE<br>Size of data buffer between interpolator and position control | |
|---|---|---|
| Default setting: 2 | Minimum input limit: 2 | Maximum input limit: 35 |
| Changes effective after POWER ON | Protection level: 3 / 2 | Unit: – |
| Data type: DWORD | Applies from SW: 6.1 | |
| Meaning: | The MD specifies the size of the data buffer between the interpolator and position control.<br>If several NCUs for rotary indexing machines/multi-spindle turning machines are connected via the NCU link, the value for all connected NCUs must be set to 3. This setting compensates for the difference in transmission time between local setpoints and set points from another NCU via the NCU link.<br>In order to ensure the same time sequences for the leading axis and the following axes when using a **lead link axis**, the size of the servo buffer on the NCU which is supplying the master value must be set to 4. The default value 2 is maintained on the NCUs that are interpolating the following axes.<br>When interpolating via NCUs with **different IPO cycles**, the maximum cycle ratio of the link cycle to IPO cycle (of the fast NCU) can be 17. Use the following formula for assigning MM_SERVO_FIFO_SIZE:<br>MM_SERVO_FIFO_SIZE = 2 * link cycle / IPO cycle +1<br>This formula applies for the NCU with the fast IPO cycle and the NCU with the slow IPO cycle. Examples:<br>NCU with fast IPO cycle:<br>Link cycle = 16 ms<br>IPO cycle n = 2 ms<br>The cycle ratio is 8, the value of the formula 2 * 8 + 1 = 17<br>The value 3 must be set on the slow NCU. | |

| 18780<br>MD number | MM_NCU_LINK_MASK<br>Activation of NCU link communication | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 3 |
| Changes effective after POWER ON | Protection level: 1 / 1 | Unit: – |
| Data type: DWORD | Applies from SW: 5 | |
| Meaning: | Activation of NCU link communication.<br>Bit-coded activation data.<br>Bit 0 must be set to activate link communication.<br>(can be used for start-up of local axes [1:1], before link connections are powered up)<br>If, in addition, enabling is to be activated for the **NCU link with different IPO and position controller cycles** (SW 6), then bit 1 must also be set. | |
| Related to .... | IS_LOCAL_LINK_AXIS, NCU_LINK_NO, LINK_TERMINATION, LINK_NUM_OF_MODULES, LINK_BAUDRATE_SWITCH, LINK_RETRY_CTR | |

| 18781<br>MD number | NCU_LINK_CONNECTIONS<br>Number of internal link connections | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 32 |
| Changes effective after POWER ON | Protection level: 0 / 3 | Unit: – |
| Data type: DWORD | Applies from SW: | |

| **18781**<br>MD number | **NCU_LINK_CONNECTIONS**<br>Number of internal link connections |
|---|---|
| Meaning: | Value = 0:<br>The software determines the internal link connections automatically<br><br>Value > 0:<br>Number of internal link connections from each NCU to every other NCU. These link connections record the non-cyclic messages. Each of these connections can transmit 240 bytes of raw data.<br>Non-cyclic messages occur in the case of alarms, container switches and link variables |
| MD irrelevant for ...... | Systems without link modules |
| Fig. | 2-17, 2-18 |
| Application examples | See Section 2.9 |
| Special cases, errors, ... ... | Total of cyclic and non-cyclic connections > 32. |
| Related to .... | No. of link axes, lead link axes, no. of NCUs, no. of link modules |
| References | |

| **18790**<br>MD number | **MM_MAX_TRACE_LINK_PIONTS**<br>Size of trace data buffer for NCU link | |
|---|---|---|
| Default setting: 8 | Minimum input limit: 0 | Maximum input limit: 500 |
| Changes effective after POWER ON | Protection level: 1 / 1 | Unit: – |
| Data type: DWORD | Applies from SW: 5 | |
| Meaning: | MM_MAX_TRACE_LINK_DATAPOINTS defines the size of an internal data buffer containing the trace recordings for the NCU link functionality.<br>The MD is evaluated only if bit 0 is set in MM_TRACE_LINK_DATA_FUNCTION. | |
| Related to .... | TRACE_SCOPE_MASK, MM_TRACE_DATA_FUNCTION,<br>MM_MAX_TRACE_DATAPOINTS,<br>TRACE_STARTTRACE_EVENT, TRACE_STARTTRACE_STEP,<br>TRACE_STOPTRACE_EVENT, TRACE_STOPTRACE_STEP,<br>TRACE_VARIABLE_NAME, TRACE_VARIABLE_INDEX,<br>MM_TRACE_LINK_DATA_FUNCTION | |

## 4.2.2 Channelspecific machine data

| **28160**<br>MD number | **MM_NUM_LINKVAR_ELEMENTS**<br>Number of write elements for the NCU link variables | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: – |
| Changes effective after POWER ON | Protection level: 7 / 2 | Unit: – |
| Data type: DWORD | Applies from SW: 5 | |
| Meaning: | Defines the number of elements available to the user for writing link variables ($A_DLx) from the parts program.<br>Each write process to a link variable requires a write element. Enabling the element is synchronous with the next traversing block. This way the MD defines the number of global link variables that can be written from a parts program block. Enabled write elements can be reused for new write processes.<br>The memory requirements per element are approx. 24 bytes. One element is required to write one link variable. The element is enabled in synchronism with the next motion block. The elements are also used for block search with NCU link variables. | |

### 4.2.3 Axis-specific machine data

| | | | |
|---|---|---|---|
| **30554**<br>MD number | **AXCONF_ASSIGN_MASTER_NCU**<br>Initial setting defining which NCU generates setpoints for the axis | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 16 | |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – | |
| Data type: BYTE | Applies from SW: 5.3 | | |
| Meaning: | The MD is evaluated only if NCU link communication is configured.<br>If a machine axis is activated via $MC_AXCONF_LOGIC_MACHAX_TAB in several NCUs of a grouping, then a master NCU must be defined. This master NCU will be assigned the task of generating setpoints for the axis during power-up.<br>In the case of axes activated in only one NCU, the number of the relevant NCU or 0 must be entered here. Any other entry will generate an alarm during power-up. | | |
| Related to .... | MD 10002: AXCONF_LOGIC_MACHAX_TAB,<br>MD 30550: AXCONF_ASSIGN_MASTER_CHAN | | |

| | | | |
|---|---|---|---|
| **30560**<br>MD number | **IS_LOCAL_LINK_AXIS**<br>Axis is a local link axis | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 0 | |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – | |
| Data type: BOOLEAN | Applies from SW: 5 | | |
| Meaning: | The MD is evaluated only if the NCU link function "$MN_MM_NCU_LINK_MASK = 1" has been activated, but NCU link communication is not yet functional, e.g. owing to the fact that not all NCUs in the NCU link grouping have yet been started up.<br>Setting IS_LOCAL_LINK_AXIS = 0 determines that the axis will not be linked to the local NCU during power-up, because this would require an external partner (another NCU).<br>Drive data can, however, be supplied to the drive for this axis. | | |
| MD irrelevant for ...... | Systems without link modules | | |
| Related to .... | MM_NCU_LINK_MASK | | |

| | | | |
|---|---|---|---|
| **32990**<br>MD number | **POSCTRL_DESVAL_DELAY_INFO**<br>Current position setpoint delay | | |
| Default setting: 0.0 | Minimum input limit: – | Maximum input limit: – | |
| Changes effective after –, read-only | Protection level: 0 / 7 | Unit: s | |
| Data type: DOUBLE | Applies from SW: 6 | | |
| Meaning: | This MD displays the position controller's additional setpoint delay with the current controller structure. With NCU link with different position controller cycles, the setting is performed **automatically** and can be changed for the entire NCU in MD 10065: POSCTRL_DESVAL_DELAY.<br>The value without feedforward control is displayed in index 0.<br>The value with speed feedforward control is displayed in index 1.<br>The value with torque feedforward control is displayed in index 2. | | |
| MD irrelevant for ...... | Systems without NCU link. | | |
| Related to .... | MD 10065: POSCTRL_DESVAL_DELAY | | |

## 4.3 Setting data for link communication

| | | | | | |
|---|---|---|---|---|---|
| **41700**<br>SD number | **AXCT_SWWIDTH**<br>Axis container rotation setting | | | | |
| Default setting: 1 | | Minimum input limit: 1 | | Maximum input limit: – | |
| Changes effective after NewConfig | | | Protection level: 3 / 3 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 5 | |
| Meaning: | Setting specifying how many increments an axis container must rotate.<br>The SD is NCU-global.<br>As an axis container in which not only local axes are administered is NCU-global, in this case SD 41700 is also NCU-global. All **NCUs** whose axes appear in the axis container use the increment value stored here.<br>All **channels** that access an axis container use the increment value stored here.<br>If a value which is higher than the number of slots occupied in the relevant axis container is entered, the value is calculated modulo in relation to the number of occupied slots.<br>Unlike other setting data, a NEWCONF must be performed to refresh this setting data. | | | | |

■

# Notes

# Signal Descriptions 5

## 5.1 Defined logical functions/defines

---

**Note**

The specified values must be entered in the interface areas indicated in the tables below. The values must be checked to distinguish between the logical functions and results.

---

Table 5-1 BUSTYP

| Name | Value | Interface DB19 | Meaning |
|---|---|---|---|
| MPI | 1 | DBW 100,102,104, 120, 130 bits 8–15 | MMC to MPI, 187.5 kbaud |
| OPI | 2 | “ | MMC to MPI, 1.5 kbaud |

STATUS

Table 5-2 Functions

| Name: Status | Value | Interface DB19 | Meaning |
|---|---|---|---|
| `OFFL_REQ_PLC` | 1 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | PLC to MMC: PLC would like to suppress MMC, sends offline request to MMC |
| `OFFL_CONF_PLC` | 2 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | MMC to PLC: Acknowledgement of `OFFL_REQ_PLC`<br>The meaning of the signal is dependent on Z_INFO DBB 125 or DBB 135 |
| `OFFL_REQ_OP` | 3 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | MMC to PLC: MMC would like to go offline from this NCU and outputs an offline request |
| `OFFL_CONF_OP` | 4 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | PLC to MMC: Acknowledgement of `OFFL_REQ_OP`<br>The meaning of the signal is dependent on Z_INFO DBB 125 or DBB 135 |
| `ONL_PERM` | 5 | Online request interface<br>DBB 108 | PLC to MMC: PLC notifies MMC as to whether it can go online or not.<br>The meaning of the signal is dependent on PAR_Z_INFO: DBB109 |
| `S_ACT` | 6 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | MMC to PLC: MMC goes online or changes operating focus.<br>The meaning of the signal is dependent on Z_INFO DBB 125 or DBB 135 |

Table 5-2 Functions

| Name: Status | Value | Interface DB19 | Meaning |
|---|---|---|---|
| `OFFL_REQ_FOC` | 7 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | MMC to PLC: MMC would like to remove operating focus from this NCU |
| `OFFL_CONF_FOC` | 8 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | PLC to MMC: Acknowledgement of `OFFL_REQ_FOC`<br>The meaning of the signal is dependent on Z_INFO DBB 125 or DBB 135 |
| `ONL_REQ_FOC` | 9 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | MMC to PLC: MMC would like to set operating focus to this NCU |
| `ONL_PERM_FOC` | 10 | Online interface<br>1. : DBB 124<br>2. : DBB 134 | PLC to MMC: Acknowledgement of `ONL_REQ_FOC`<br>The meaning of the signal is dependent on Z_INFO DBB 125 or DBB 135 |

Table 5-3 Z_INFO

| Name | Value | Interface DB19 | Meaning |
|---|---|---|---|
| DISC_FOC | 9 | DBB125<br>DBB135 | MMC switches operating focus to another NCU. |
| OK | 10 | DBB 109 bits 0–3<br>DBB125<br>DBB135 | Positive acknowledgement |
| CONNECT | 11 | DBB125<br>DBB135 | MMC has gone online on this NCU. |
| MMC_LOCKED | 13 | DBB 109 bits 0–3<br>DBB125<br>DBB135 | MMC has set switchover disable.<br>Processes which must not be interrupted by a switchover operation are currently in progress on this MMC. |
| PLC_LOCKED | 14 | DBB 109 bits 0–3<br>DBB125<br>DBB135 | The MMC switchover disable is set in the MMC-PLC interface. MMC cannot go offline from this NCU or change operating focus. |
| PRIO_H | 15 | DBB 109 bits 0–3<br>DBB125<br>DBB135 | MMCs with a higher priority are operating on this NCU. MMC cannot go online to this NCU. |

Table 5-4 STATUS and Z_INFO can be combined as follows

| Name: Status | Z_INFO | Meaning |
|---|---|---|
| OFFL_REQ_PLC | OK | PLC would like to suppress online MMC and sends the offline request. |
| OFFL_CONF_PLC | OK | MMC positively acknowledges the offline request from the PLC.<br>The MMC then goes offline. |
| OFFL_CONF_PLC | MMC_LOCKED | MMC outputs negative acknowledgement to offline request.<br>MMC does not go offline because uninterruptible processes are in progress on the MMC. |
| OFFL_REQ_OP | OK | MMC would like to go offline from the online NCU and outputs an offline request. |
| OFFL_CONF_OP | OK | PLC positively acknowledges the offline request.<br>The MMC then goes offline from this NCU. |
| OFFL_CONF_OP | PLC_LOCKED | PLC outputs negative acknowledgement to offline request from MMC.<br>User has set the MMC switchover disable, MMC cannot go offline, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, 1st or 2nd MMC-PLC interface. |
| ONL_PERM | No. of MMC-PLC online interface, OK | PLC issues the online enabling command to the requesting MMC. MMC can then go online to this NCU. Contents of Z_INFO:<br>Bit 0 ..3: OK<br>Bit 4 .. 7: No. of the MMC-PLC online interface to which MMC must be connected:<br>1 First MMC-PLC online interface<br>2 Second MMC-PLC online interface |
| ONL_PERM | MMC_LOCKED | The requesting MMC cannot go online.<br>Two MMCs on which uninterruptible processes are in progress are connected online to this NCU. The PLC cannot suppress either of the two MMCs. |

Table 5-4 STATUS and Z_INFO can be combined as follows

| Name: Status | Z_INFO | Meaning |
|---|---|---|
| ONL_PERM | PLC_LOCKED | The requesting MMC cannot go online.<br>User has set the MMC switchover disable,<br>MMCx_SHIFT_LOCK = TRUE,<br>x=1 or 2, 1st or 2nd MMC online interface. |
| ONL_PERM | PRIO_H | The requesting MMC cannot go online.<br>Two MMCs that are both higher priority than the requesting MMC are connected online to the NCU.<br>The PLC cannot suppress either of the two MMCs. |
| S_ACT | CONNECT | The requesting MMC has gone online.<br>The PLC now switches on the MMC sign-of-life monitoring function. |
| S_ACT | DISC_FOCUS | Server MMC has disconnected the operating focus from this NCU. |
| OFFL_REQ_FOC | OK | Server MMC would like to disconnect the operating focus from this NCU and outputs an offline focus request. |
| OFFL_CONF_FOC | OK | PLC positively acknowledges the focus offline request.<br>Server MMC can disconnect the operating focus. |
| OFFL_CONF_FOC | PLC_LOCKED | PLC negatively acknowledges the offline focus request.<br>User has set the MMC switchover disable, server MMC cannot disconnect the operating focus,<br>MMCx_SHIFT_LOCK = TRUE.<br>x=1 or 2, 1st or 2nd MMC-PLC interface. |
| ONL_REQ_FOC | OK | Server MMC would like to set the operating focus on this NCU and outputs an online focus request. |
| ONL_PERM_FOC | OK | PLC positively acknowledges the online focus request.<br>Server MMC then switches the operating focus to this NCU. |
| ONL_PERM_FOC | PLC_LOCKED | PLC negatively acknowledges the online focus request.<br>User has set the MMC switchover disable, server MMC cannot set the operating focus, MMCx_SHIFT_LOCK = TRUE,<br>x=1 or 2, 1st or 2nd MMC-PLC interface. |

## 5.2 Interfaces in DB 19 for M:N

The MMC-PLC interface in DB19 is divided into 3 areas

**Online request interface**

The online request sequence is executed on this interface if an MMC wants to go online.

MMC 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 MMC sends its parameters and waits for online permission (in PAR_STATUS, PAR_Z_INFO).

MMC parameter transfer:

| | |
|---|---|
| Client identification | –> PAR_CLIENT_IDENT |
| MMC type | –> PAR_MMC_TYP |
| MCP address | –> PAR_MSTT_ADR |

With the positive online permission, the PLC also sends the number of the MMC-PLC online interface DBB109.4-7 to be used by the MMC.

The MMC then goes online and occupies the online interface assigned by the PLC.

**Online interfaces**

Two MMCs can be connected online to one NCU at the same time.

The online interface is available for each of the two online MMCs separately.

After a successful online request sequence, the MMC receives the number of its online interface from the PLC.

The MMC parameters are then transferred to the corresponding online interface by the PLC.

The MMC goes online and occupies its own online interface via which data are then exchanged between the MMC and PLC.

**MMC data interfaces**

User data from/to the MMC are defined on these:

– DBB 0–49 MMC 1 interface

– DBB 50–99 MMC 2 interface

These data and signals are always needed to operate MMCs.

**M:N sign-of-life monitoring**

This is an additional monitoring function which must not be confused with the MMC sign-of-life monitor. For further information, please refer to the relevant signals.

In certain operating states, MMCs 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:
MMC with an activated control unit switchover function must be capable of starting up an NCU without issuing an online request first.

MMC 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 an MMC issues an online request, it must check the sign of life to establish whether the M:N switchover is activated in the PLC.

Procedure:

MMC reads the sign of life at instants T0 and T0 + 1.

Case 1: Negative acknowledgement for read operation, DB19 does not exist. MMC goes online without prior online request.

Case 2: m_to_n_alive = 0
Control unit switchover not activated.
MMC goes online without prior online request.

Case 3: m_to_n_alive (T0) = m_to_n_alive (T0+1)
Control unit switchover not activated
MMC goes online without prior online request.

Case 4: m_to_n_alive (T0) <> m_to_n_alive (T0+1)
Control unit switchover activated

Cases 1 to 3 apply only under special conditions and not in normal operation.

## Online request interface

| **No.** | **Name** |
|---|---|
| Possible value | Meaning |

| DB19 DBW100 | ONL_REQUEST |
|---|---|
| Client_Ident | MMC would like to go online and use the online request interface. It first writes its Client_Ident as a request.<br><br>Bit 8 .. 15: Bus type:<br>MPI 1 or<br>MCP 2<br>Bit 0 .. 7: MMC bus address |

| DB19 DBW102 | ONL_CONFIRM |
|---|---|
| Client_Ident | If the online request interface is not being used by another MMC, the PLC returns the Client identification as positive acknowledgement.<br>Bit 8 .. 15: Bus type:<br>MPI 1 or<br>MCP 2<br>Bit 0 .. 7: MMC bus address |

| DB19 DBW104 | PAR_CLIENT_IDENT MMC parameter transfer to PLC |
|---|---|
| Client_Ident | Bit 8 .. 15: Bus type:<br>MPI 1 or<br>OPI 2<br>Bit 0 .. 7: MMC bus address |

| DB19 DBB106 | PAR_MMC_TYP MMC parameter transfer to PLC |
|---|---|
| MMC type from NETNAMES.INI | Type properties of the MMC 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 MMC parameter transfer to PLC |
|---|---|
| MCP address from NETNAMES.INI | Address of MCP to be switched over or activated/deactivated with the MMC. Parameter from NETNAMES.INI |
| 255 | No MCP is assigned to MMC, no MCP will be activated/deactivated |

| DB19 DB108 | PAR_STATUS PLC sends MMC pos./neg. online permission |
|---|---|
| ONL_PERM (5) | PLC notifies MMC as to whether it can go online or not. The meaning of the signal is dependent on PAR_Z_INFO: |

| DB19 DBB109 | PAR_Z_INFO PLC sends MMC pos./neg. online permission |
|---|---|
| No. of MMC-PLC online interface, OK (10) | PLC issues the online enabling command to the requesting MMC.<br>MMC can then go online to this NCU.<br>Bit 0 ..3: OK<br>Bit 4 .. 7: No. of the MMC-PLC online interface to which MMC must be connected:<br>1 First MMC-PLC online interface<br>2 Second MMC-PLC online interface |
| MMC_LOCKED (13) | The requesting MMC cannot go online. Two MMCs on which uninterruptible processes are in progress are connected online to this NCU. The PLC cannot displace either of the two MMCs. |
| PLC_LOCKED (14) | The MMC switchover disable is set in the MMC-PLC interface. |
| PRIO_H (15) | The requesting MMC cannot go online. Two MMCs that are both higher priority than the requesting MMC are connected online to the NCU. The PLC cannot displace either of the two MMCs. |

### 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 MMCs that the M:N switchover is active and ready. |

### 1. MMC-PLC online interface

| DB19 DBW120 | MMC1_CLIENT_IDENT |
|---|---|
| | See PAR_CLIENT_IDENT<br>After issuing positive online permission, the PLC transfers the MMC parameters to the online interface PAR_CLIENT_IDENT –> MMC1_CLIENT_IDENT |

| DB19 DBB122 | MMC1_TYP |
|---|---|
| | See PAR_MMC_TYP<br>After issuing positive online permission, the PLC transfers the MMC parameters to the online interface PAR_MMC_TYP –> MMC1_TYP |

| DB19 DBB123 | MMC1_MSTT_ADR |
|---|---|
| | See PAR_ MSTT_ADR<br>After issuing positive online permission, the PLC transfers the MMC parameters to the online interface PAR_ MSTT_ADR –> MMC1_MSTT_ADR |

| DB19 DBB124 | MMC1_STATUS<br>Requests from online MMC to PLC or vice versa<br>The meaning of the signal is dependent on MMC1_Z_INFO<br>see also:<br>DEFINEs possible combinations of STATUS and Z_INFO for control unit switchover |
|---|---|
| OFFL_REQ_PLC (1) | PLC to MMC: PLC would like to suppress MMC, sends offline request to MMC |
| OFFL_CONF_PLC (2) | MMC to PLC: Acknowledgement of OFFL_REQ_PLC |
| OFFL_REQ_OP (3) | MMC to PLC: MMC would like to go offline from this NCU and outputs an offline request |
| OFFL_CONF_OP (4) | PLC to MMC: Acknowledgement of OFFL_REQ_OP |
| S_ACT (6) | MMC to PLC: MMC goes online or changes operating focus |
| OFFL_REQ_FOC (7) | MMC to PLC: MMC would like to take operating focus away from this NCU |
| OFFL_CONF_FOC (8) | PLC to MMC: Acknowledgement of OFFL_REQ_FOC |
| ONL_REQ_FOC (9) | MMC to PLC: MMC would like to set operating focus to this NCU |
| ONL_PERM_FOC (10) | PLC to MMC: Acknowledgement of ONL_REQ_FOC |

| DB19 DBB125 | MMC1_Z_INFO<br>Requests from online MMC to PLC or vice versa<br>The meaning of the signal is dependent on MMC1_STATUS,<br>see also:<br>DEFINEs possible combinations of STATUS and Z_INFO<br>for control unit switchover |
|---|---|
| DISC_FOC (9) | MMC switches operating focus to another NCU. |
| OK (10) | Positive acknowledgement |
| CONNECT (11) | MMC has gone online to this NCU. |
| PLC_LOCKED (14) | The MMC switchover disable is set in the MMC-PLC interface. MMC cannot go offline from this NCU or change operating focus. |
| PRIO_H (15) | MMCs with higher priority are online to this NCU. MMC cannot go online to this NCU. |

## Bit signals

| **DB 19**<br>**DBX 126.0**<br>Data Block | **MMC1_SHIFT_LOCK**<br>Disable/enable MMC switchover | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ––> 1 | MMC switchover or change in operating focus is disabled.<br>The current MMC-NCU connection status remains unchanged. | |
| Signal state 0 or signal transition 1 ––> 0 | MMC switchover or change in operating focus is enabled. | |

| **DB 19**<br>**DBX 126.1**<br>Data Block | **MMC1_MSTT_SHIFT_LOCK**<br>Disable/enable MCP switchover | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ––> 1 | MCP switchover is disabled.<br>The current MCP-NCU constellation remains unchanged. | |
| Signal state 0 or signal transition 1 ––> 0 | MCP switchover is enabled | |

| **DB 19**<br>**DBX 126.2**<br>Data Block | **MMC1_ACTIVE_REQ**<br>MMC 1 requests active operating mode | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ––> 1 | MMC to PLC: Passive MMC 1 requests active operating mode | |
| Signal state 0 or signal transition 1 ––> 0 | PLC to MMC: Request received | |

| **DB 19**<br>DBX 126.3<br>Data Block | **MMC1_ACTIVE_PERM**<br>Active/passive operating mode | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ——> 1 | PLC to MMC:<br>Passive MMC can change to active operating mode | |
| Signal state 0 or signal transition 1 ——> 0 | PLC to MMC:<br>Active MMC must change to passive operating mode | |

| **DB 19**<br>**DBX 126.4**<br>Data Block | **MMC1_ACTIVE_CHANGED**<br>Active/passive operating mode of MMC | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ——> 1 | MMC to PLC:<br>MMC has completed changeover from passive to active mode | |
| Signal state 0 or signal transition 1 ——> 0 | MMC to PLC:<br>MMC has completed changeover from active to passive mode | |

| **DB 19**<br>**DBX126.5**<br>Data Block | **MMC1_CHANGE_DENIED**<br>Operating mode changeover rejected | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ——> 1 | MMC to PLC or PLC to MMC depending on interface:<br>Operating mode cannot be changed owing to uninterruptible processes on active MMC | |
| Signal state 0 or signal transition 1 ——> 0 | MMC to PLC or PLC to MMC depending on interface:<br>Acknowledgement of MMC1_CHANGE_DENIED (FALSE —>TRUE) | |

**2. MMC–PLC online interface**

The signals of the 2nd MMC-PLC online interface are analogous in meaning to the signals of the 1st MMC-PLC online interface. MMC2_ ... replaces MMC1_... in the explanatory texts.

| | |
|---|---|
| DB19 DBW130 | MMC2_CLIENT_IDENT |
| | See DB19 DBW120 |

| | |
|---|---|
| DB19 DBB132 | MMC2_TYP |
| | See DB19 DBW122 |

| | |
|---|---|
| DB19 DBB133 | MMC2_MSTT_ADR |
| | See DB19 DBB 123 |

| | |
|---|---|
| DB19 DBB134 | MMC2_STATUS |
| | See DB19 DBB124 |

| | |
|---|---|
| DB19 DBB135 | MMC2_Z_INFO |
| | See DB19 DBB 125 |

| | |
|---|---|
| DB19 DBX136.0 | MMC2_SHIFT_LOCK |
| | See DB19 DBX126.0 |

| | |
|---|---|
| DB19 DBX136.1 | MMC2_MSTT_SHIFT_LOCK |
| | See DB19 DBX126.1 |

| | |
|---|---|
| DB19 DBX136.2 | MMC2_ACTIVE_REQ |
| | See DB19 DBX126.2 |

| | |
|---|---|
| DB19 DBX136.3 | MMC2_ACTIVE_PERM |
| | See DB19 DBX 126.3 |

| | |
|---|---|
| DB19 DBX136.4 | MMC2_ACTIVE_CHANGED |
| | See DB19 DBX 126.4 |

| | |
|---|---|
| DB19 DBW136.5 | MMC2_CHANGE_DENIED |
| | See DB19 DBX126.5 |

**MMC sign-of-life monitor**

After an MMC has gone online to an NCU, the MMC sign of life is set in the interface. (E_BTSSReady, E_MMCMPI_Ready, E_MMC2Ready)

The signals are automatically set by the MMC when it goes online and stay set for as long as it remains online.

They are provided separately for each MMC-PLC interface and used by the PLC to monitor the MMC sign of life.

**First MMC-PLC online interface**

A distinction between an MMC 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 MMC is online.

| DB**10** DBX104.0 | MCP1 ready |
|---|---|
| FALSE | MCP1 is not ready |
| TRUE | MCP1 is ready |

| DB**10** DBX104.0 | MCP2 ready |
|---|---|
| FALSE | MCP2 is not ready |
| TRUE | MCP2 is ready |

| DB**10** DBX104.2 | HHU ready |
|---|---|
| FALSE | HHU is not ready |
| TRUE | HHU is ready |

| DB**10** DBX108.3 | E_MMCBTSSReady |
|---|---|
| FALSE | No MMC online to OPI |
| TRUE | MMC online to OPI |

| DB**10** DBX108.2 | E_MMCMPIReady |
|---|---|
| FALSE | No MMC online to MPI |
| TRUE | MMC online to MPI |

**Second MMC-PLC online interface**

This interface utilizes a group signal for both bus types. No distinction is made between OPI and MPI.

| DB**10** DBX108.1 | E_MMC2Ready |
|---|---|
| FALSE | No MMC online to OPI or MPI |
| TRUE | MMC online to OPI or MPI |

The sign-of-life monitor is switched on by the PLC as soon as an MMC has gone online to its interface and switched off again when it goes offline.

Sign-of-life monitor is **switched on**:

– As soon as an MMC logs on online to its MMC-PLC interface with S_ACT/ CONNECT.

Sign-of-life monitor is **switched off**:

– As soon as MMC goes offline

1. MMC wants to switch over and logs off from the PLC with OFFL_REQ_OP/ OK
   PLC acknowledges the MMC with OFFL_CONF_OP/ OK
2. MMC is suppressed by the PLC with OFFL_REQ_PLC/ OK
   MMC acknowledges the PLC with OFFL_CONF_PLC/ OK

In both instances the PLC detects that an MMC is going offline and waits for the TRUE–FALSE edge of its sign-of-life signal.

The PLC then ceases to monitor the sign-of-life signal.

## 5.3 Signals for NCU link and axis container

| | | |
|---|---|---|
| **DB 10**<br>**DBX 107.6**<br>Data Block | **NCU link active**<br><br>Signal from NC channel –> PLC | |
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ––> 1 | NCU link communication is active | |
| Signal state 0 or signal transition 1 ––> 0 | No NCU link communication is active | |
| Signal irrelevant for ...... | Systems without NCU link modules | |
| References | PHD | |

| | | |
|---|---|---|
| **DB 31-61**<br>**DBX 60.1**<br>Data Block | **NCU link axis active**<br><br>Signal from NC axis –> PLC | |
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ––> 1 | Axis is active as NCU link axis | |
| Signal state 0 or signal transition 1 ––> 0 | Axis is used as a local axis | |
| Signal irrelevant for ...... | Systems without NCU link modules | |
| References | PHD | |

| | | |
|---|---|---|
| **DB 31-61**<br>**DBX 61.1**<br>Data Block | **Axis container rotation active**<br><br>Signal from NC axis –> PLC | |
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 5 |
| Signal state 1 or signal transition 0 ––> 1 | An axis container rotation is active for the axis | |
| Signal state 0 or signal transition 1 ––> 0 | An axis container rotation is not active for the axis | |

| | | |
|---|---|---|
| **DB 31-61**<br>**DBX 61.2**<br>Data Block | **Axis ready**<br><br>Signal from NC axis –> PLC | |
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 5 |
| Meaning | The signal is processed on the home NCU in the NCU link grouping.<br>The home NCU is the NCU to which the axis is physically connected. | |
| Signal state 1 or signal transition 0 ––> 1 | Axis is ready | |
| Signal state 0 or signal transition 1 ––> 0 | Axis is not ready<br>This status is set if<br>– the channel or<br>– the operating mode group or<br>– the NCK<br>has generated the “not ready” alarm. | |

■

# Examples 6

## 6.1 Configuration file NETNAMES.INI with control unit management option

A sample configuration file NETNAMES.INI for the MMC 1 control unit for a system with four NCUs on the OPI is outlined below.
See Subsection 2.1.4 for explanations.

---

**Note**

The marginal notes (bold print) on the left of the page serve to structure the information and are not part of the file.

---

; NETNAMES.INI Example 1 start

**MMC identification**

```
; Identification entry
 [own]
 owner          = MMC_1
```

**MMC-to-NCU connections**

```
; Connection entry
[conn MMC_1]
conn_1          = NCU_1          ; NCU 1
conn_2          = NCU_2          ; NCU 2
conn_3          = NCU_3          ; NCU 3
conn_4          = NCU_4          ; NCU 4
```

**Bus identification**

```
; Descriptive entry
[param network]
bus             = OPI            ; OPI bus (1.5 Mbaud)
```

**MMC description**

```
[param MMC_1]
mmc_typ        = 40          ; = 0100 0000: MMC is server and
                             ; main control panel
mmc_bustyp     = BTSS        ; bus the MMC is attached to
mmc_address    = 10          ; MMC address
mstt_address   = 6           ; address of MCP to be switched simult-
                             ; aneously
name           = MMC_LINKS   ; Name of MMC
start_mode     = ONLINE      ; MMC switches online to the DEFAULT
                             ; NCU during booting according to
                             ; channel data, see below
```

**Description of NCU components**

```
[param NCU_1]
type           = NCU_572     ; NCU type
nck_address    = 20          ; address j of NCU component on bus
plc_address    = 20          ; address p of PLC component on bus
name           = NCU1        ; name of NCU

[param NCU_2]
type           = NCU_572     ; NCU type
nck_address    = 21          ; address j of NCU component on bus
plc_address    = 21          ; address p of PLC component on bus
name           = NCU2        ; name of NCU

[param NCU_3]
type           = NCU_572     ; NCU type
nck_address    = 22          ; address j of NCU component on bus
plc_address    = 22          ; address p of PLC component on bus
name           = NCU3        ; name of NCU

[param NCU_4]
type           = NCU_572     ; NCU type
nck_address    = 23          ; address j of NCU component on bus
plc_address    = 23          ; address p of PLC component on bus
name           = NCU4        ; name of NCU

; End of descriptive entry
```

**Channel data**

```
; Sample of a channel menu configuration
; with M:N assignment option

[chan MMC_1]
DEFAULT_logChanSet = G_1        ; group setting during power-up
DEFAULT_logChan = K_1_1         ; channel setting during power-up
ShowChanMenu  = TRUE            ; display channel menu

                  ; List of channel groups
logChanSetList    = G_1, G_2, G_3, G_4

[G_1]
logChanList       = K_1_1, K_1_2  ; Group G_1 channels

[G_2]
logChanList       = K_2_1, K_2_2  ; Group G_2 channels

[G_3]
logChanList       = K_3_1, K_3_2  ; Group G_3 channels

[G_4]
logChanList       = K_4_1, K_4_2  ; Group G_4 channels

[K_1_1]
logNCName         = NCU_1         ; 1st channel of 1st group
ChanNum           = 1

[K_1_2]
logNCName         = NCU_1         ; 2nd channel of 1st group
ChanNum           = 2

[K_2_1]
logNCName         = NCU_2         ; 1st channel of 2nd group
ChanNum           = 1

[K_2_2]
logNCName         = NCU_2         ; 2nd channel of 2nd group
ChanNum           = 2

[K_3_1]
logNCName         = NCU_3         ; 1st channel of 3rd group
ChanNum           = 1

[K_3_2]
logNCName         = NCU_3         ; 2nd channel of 3rd group
ChanNum           = 2

[K_4_1]
logNCName         = NCU_4         ; 1st channel of 4th group
ChanNum           = 1

[K_4_2]
logNCName         = NCU_4         ; 2nd channel of 4th group
ChanNum           = 2


; NETNAMES.INI example 1 end
```

**Note**

You will find further examples in the subsection entitled **Quick installation guide**.

## 6.2 User-specific re-configuring of PLC program control unit switchover

**Introduction**

The solution outlined roughly below should be selected only if at least one of the following configuring requirements is applicable:

- Displacement strategy which differs from standard functionality
- Operating mode switchover which differs from standard functionality
- Independent handling of override switch for switchover of control unit
- Existence of a 2nd machine control panel on an MMC

Method of description:

1. Description of operational sequences
2. Description of available functionality (Defines)
3. Graphic representation of sequences in diagrammatic form

Implementation details can also be obtained from the standard configuration which is included in the toolbox.

### 6.2.1 Description of operational sequences (overview)

**Overview:**

**MMC call waiting**

An MMC would like to link up with an NCU and sends this request to the PLC of the relevant NCU.

**MMC coming**

An MMC goes online to an NCU, i.e. it links up to the NCU.

**MMC going**

An MMC breaks off the link to an NCU.

**Forced break**

An MMC must abort the link with an NCU because another MMC wants to go online to the same NCU.

**Operating focus changeover to server mode**

A server maintains a permanent link to the NCUs to which it is assigned. The operator can switch the operating focus from one NCU to another without interrupting the existing link.

**Active/passive operating mode**

An online MMC can operate in two different modes:

Active mode: Operator can control and monitor

Passive mode: Operators sees header information and the “passive” identifier.

**MCP switchover**

As an option, an MCP assigned to the MMC can be switched over at the same time as the MMC.

### 6.2.2 Description of operational sequences (details)

**Introduction**

The operational sequences are described using identifiers for defined, logical functions (example: `OFFL_REQ_OP/ OK`) whose programming application has been described earlier in this section. The functions are coded according to Section 5.1. The functions store values in the interface which can be addressed from the PLC and the MMC. An MMC utilizes the online-request interface while it competing for the use of an online interface. MMCs which are already linked to an NCU utilize one of the two available online interfaces. Details of these interfaces can be found in Chapter 5 and in

**References:** Lists

are programmed.

In order to illustrate complete operating sequences, the description covers MMC activities which cannot be influenced as well as **modifiable PLC activities**.

**MMC call waiting**

If the MMC is already linked online to an NCU (online NCU) and would like to communicate with another NCU (target NCU), it must first notify the PLC of the online NCU that it wishes to switch over to the target NCU.

It sends the offline request `OFFL_REQ_OP/ OK` to the online PLC.

`OFFL_CONF_OP/ OK`:

Online PLC has received the offline request. MMC can now send an online request to the target PLC.

`OFFL_CONF_OP/ PLC_LOCKED`

Online PLC has received the offline request. The MMC switchover is disabled in the MMC-PLC interface. The MMC cannot link up with another NCU and must remain online.

On receipt of the positive acknowledgement `OFFL_CONF_OP/ OK`, the MMC sends its online request to the target PLC of the relevant NCU by transmitting its *client identification*.

*Client identification*: Unique MMC identifier comprising bus type and MMC bus address. (ONL_REQUEST DB19, DBW100)

The target PLC sends the MMC a positive or negative acknowledgement:

Pos. acknowledgement:
Target PLC returns the client identification to the MMC. (ONL_CONFIRM, DB19, DBW102)
MMC sets its parameters on the online-request interface. (Client ident, MMC type, MCP address).
MMC can go online once it has received online permission from the target PLC.

Neg. acknowledgement:
Target PLC does not return the client identification to the MMC. (ONL_CONFIRM, DB19, DBW102 not identical to client identification of requesting MMC). MMC cannot go online.

Example:
Another MMC is currently switching over to the same NCU. This switchover operation must not be interrupted. The MMC remains online to the online NCU.

Once the MMC has received positive acknowledgement from the PLC, it may need to displace another online MMC. It will then receive positive/negative online permission from the PLC.

Positive:

`ONL_PERM/ OK`

On receipt of positive online permission (DB 19, DBB 108, 109), the MMC can go online. An MMC-PLC interface is allocated to the MMC at the same time as online permission (1 or 2, details can be found in the interface description in Chapter 5).

Negative:

`ONL_PERM/ MMC_LOCKED`

The requesting MMC cannot go online. Two MMCs on which uninterruptible processes are in progress are connected online to this NCU. The PLC cannot displace either of the two MMCs. The MMC remains online to the online NCU.

`ONL_PERM/ PLC_LOCKED`

The requesting MMC cannot go online. The MMC switchover is disabled in the MMC-PLC interface. The MMC remains online to the online NCU.

`ONL_PERM/ PRIO_H`

The requesting MMC cannot go online. Two MMCs that are both higher priority than the requesting MMC are connected online to the NCU. The PLC cannot displace either of the two MMCs. The MMC remains online to the online NCU.

**MMC coming**

Once the MMC has sent an online request to the target PLC and received online permission from it, it can set up a link to the target NCU.

It goes online and notifies the PLC with (station active) `S_ACT/ CONNECT` that it has linked up with the NCU.

The MMC sets up its sign of life signal in accordance with the allocated interface.

The MMC then requests

| | |
|---|---|
| in the case of operator panel front: | Active operating mode on the target NCU or |
| in the case of a server: | Operating focus on the target NCU. |

The PLC then activates MMC sign-of-life monitoring for the new MMC.

See: Active/passive operating mode:

See: Operating focus changeover to server mode

**MMC going**

An MMC aborts communication with an NCU.

Communication can be aborted for two different reasons:

1. The operator wishes to switch the MMC to another NCU. The MMC has sent an online request to the target PLC and received online permission (`ONL_PERM/ OK`). It has notified the online PLC of its intention to switch over with `OFFL_REQ_OP/ OK` and received a positive acknowledgement (`OFFL_CONF_OP/ OK`). Due to the switchover to the target NCU, the MMC sign of life in the online PLC is changed from TRUE to FALSE. The falling edge combined with the sequence described above signals to the online PLC that the MMC has broken off the link to the online NCU. If an MCP is assigned to the MMC and activated, it is now deactivated by the PLC. Passive operating mode is set in the PLC for the MMC which has gone offline.
   See: Active/passive operating mode:
2. The MMC is excluded (displaced) from the PLC by the online request from another MMC. See displacement.

**Forced break**

Two MMCs are linked online to an NCU, each is occupying an MMC-PLC interface. A third MMC would like to go online.

The PLC must displace one of the two MMCs according to a predefined strategy.

It requests the MMC to be displaced to abort communication with the NCU by sending it the offline request (`OFFL_REQ_OP/ OK`).

The MMC returns a positive or negative acknowledgement to the PLC:

Positive:

`OFFL_CONF_PLC/ OK`

MMC breaks off the link to the NCU and switches to the offline state.

The MMC sign of life in the PLC changes from TRUE to FALSE.

The falling edge combined with the sequence described above signals to the online PLC that the MMC has broken off the link to the online NCU.

If an MCP is assigned to the MMC and activated, it must now be deactivated by the PLC.

The PLC also ceases to monitor the MMC sign of life signal.

Passive operating mode is set in the PLC for the MMC which has been displaced.

See “Active/passive operating mode” further below.

Negative:

OFFL_CONF_PLC/ MMC_LOCKED

Processes which must not be interrupted are running on the MMC (e.g.: operation via RS232C, data exchange between NCU and MMC).

The MMC remains online to the current NCU.

**Operating focus changeover to server mode**

A server maintains a permanent link to the NCUs to which it is assigned. The operator can switch the operating focus from one NCU to another without interrupting the existing link.

If the operator wishes to switch the operating focus to another NCU, the focus PLC and target PLC must first be interrogated to determine whether they will permit a focus switchover.

The MMC first sends the focus offline request signal (OFFL_REQ_FOC/ OK) to the focus PLC.

The focus PLC returns either a positive or negative acknowledgement to the MMC:

Positive:

OFFL_CONF_FOC/ OK

PLC positively acknowledges the offline focus request. MMC can disconnect the operating focus.

Negative:

OFFL_CONF_FOC/ PLC_LOCKED

PLC negatively acknowledges the online focus request. The operating focus changeover is disabled in the MMC-PLC interface (same signal as for MMC switchover disable). The operating focus remains on the current NCU.

After a positive acknowledgement (OFFL_CONF_FOC/ OK) from the focus PLC, the MMC sends query signal ONL_REQ_FOC/ OK regarding focus changeover to the target PLC.

The target PLC sends the MMC a positive or negative acknowledgement:

Positive:

ONL_PERM_FOC / OK

PLC positively acknowledges the offline focus request. MMC can disconnect the operating focus.

Negative:

ONL_PERM_FOC / PLC_LOCKED

PLC negatively acknowledges the online focus request. The operating focus changeover is disabled in the MMC-PLC interface (same signal as for MMC switchover disable). The operating focus remains on the current NCU.

After the MMC has received permission from the target PLC to change the operating focus (ONL_PERM_FOC, OK), the MMC logs off from the focus PLC with S_ACT/ DISC_FOCUS and changes the focus to the target PLC.

The MMC must finally request active operating mode in the target NCU. The previous focus PLC must set active operating mode for this MMC-PLC interface after receiving S_ACT/ DISC_FOCUS and deactivate any active MCP assigned to the MMC which has gone offline.

See: Active/passive operating mode

**Active/passive operating mode:**

After an MMC has gone online to an NCU, it can assume one of two different operating states:

Active mode: Operator can control and monitor

Passive mode: Operator sees header information and the "passive" status identifier.

After switching to an NCU, it first requests active operating mode in the online PLC.

If two MMCs are linked online simultaneously to an NCU, one of the two is always in active mode and the other in passive mode.

The operator can request active mode on the passive MMC at the press of a button.

If an MCP has been configured for the online MMCs, the MCP of the active MMC is switched on.

The MCP of the passive MMC is deactivated, i.e. only one MCP is active at a time on an NCU.

Four signals are provided in the MMC-PLC interface for each of the two online MMCs. These signals are used by the PLC to control operating mode changeovers.

Table 6-1 Signals (x = 1, 2: 1. 1st or 2nd MMC-PLC interface)

| MMC-PLC interface | Value | Meaning |
|---|---|---|
| MMCx_ACTIVE_REQ | FALSE–>TRUE | MMC to PLC: Passive MMC requests active operating mode |
| | TRUE–>FALSE | PLC to MMC: Request received |
| MMCx_ACTIVE_PERM | FALSE–>TRUE | PLC to MMC: Passive MMC can change to active operating mode |
| | TRUE–>FALSE | PLC to MMC: Active MMC must change to passive operating mode |
| MMCx_ACTIVE_CHANGED | FALSE–>TRUE | MMC to PLC: MMC has completed changeover from passive to active mode |
| | TRUE–>FALSE | MMC to PLC: MMC has completed changeover from active to passive mode |
| MMCx_CHANGE_DENIED | FALSE–>TRUE | MMC to PLC or PLC to MMC depending on interface: Operating mode cannot be changed owing to uninterruptible processes on active MMC |
| | TRUE–>FALSE | MMC to PLC or PLC to MMC depending on interface: Acknowledgement of MMCx_CHANGE_DENIED(FALSE–>TRUE) |

An example of how operating modes can be switched over is described in the following sequence.

Two MMCs online to one NCU, MMC_1 in active operating mode, MMC_2 in passive operating mode, operator requests active operating mode on MMC_2.

These sequence applies equally to the following cases:

– An MMC goes online to an NCU to which another MMC is linked online and in active mode. It requests active operating mode.

– An MMC goes online to an NCU to which no other MMC is linked online. It requests active operating mode. (The sequence PLC requests active MMC to switch to passive operating mode is not included here).

Signal state for this case:

Table 6-2

| MMC_1 | VALUE | MMC_2 | Value |
|---|---|---|---|
| MMC1_ACTIVE_REQ | FALSE | MMC2_ACTIVE_REQ | FALSE |
| MMC1_ACTIVE_PERM | TRUE | MMC2_ACTIVE_PERM | FALSE |
| MMC1_ACTIVE_CHANGED | TRUE | MMC2_ACTIVE_CHANGED | FALSE |
| MMC1_CHANGE_DENIED | FALSE | MMC2_CHANGE_DENIED | FALSE |

MMC_2 requests active operating mode and sets MMC_2_AKTIVE_REQ = TRUE.

The PLC acknowledges the request from MMC_2 with MMC_2_ACTIVE_REQ = FALSE.

The PLC then requests MMC_1 to change to passive operating mode with MMC1_ACTIVE_PERM = FALSE.

Communication can be aborted for two different reasons:

1. MMC_1 can change to passive operating mode:
   MMC_1 switches from active to passive operating mode and acknowledges the changeover with
   MMC1_ACTIVE_CHANGED = FALSE.
   If an MCP is assigned to the MMC and activated, it is now deactivated by the PLC.
   The PLC notifies MMC_2 that it can change to active operating mode by sending MMC2_ACTIVE_PERM = TRUE.
   MMC_2 changes state and acknowledges the change with
   MMC2_ACTIVE_CHANGED = TRUE. If an MCP is assigned to MMC_2, it is now activated by the PLC.
2. MMC_1 cannot change to passive mode (processes which cannot be interrupted are in progress on MMC_1):
   MMC_1 sets MMC1_CHANGE_DENIED = TRUE, operating status cannot be changed.
   The PLC acknowledges with MMC1_CHANGE_DENIED = FALSE and grants MMC_1 permission to remain in active mode with
   MMC1_ACTIVE_PERM = TRUE. By sending MMC2_CHANGE_DENIED = TRUE, it notifies MMC_2 that MMC_1 cannot switch over to passive mode.
   MMC_2 then acknowledges with MMC2_CHANGE_DENIED = FALSE and remains in passive operating mode.

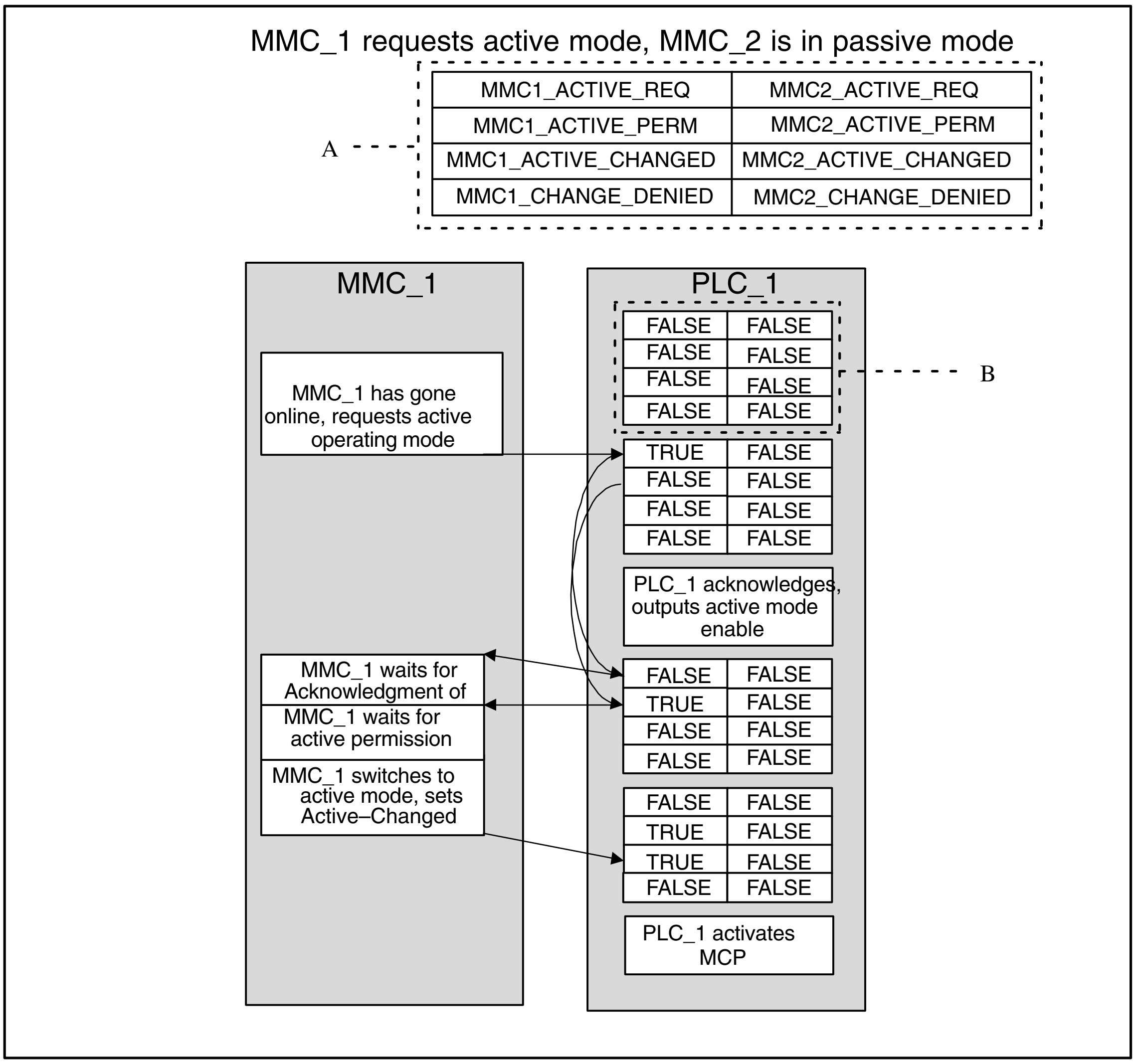



Fig. 6-1 MMC_1 requests active mode, MMC_2 is in passive mode

**Please note**

The arrangement of the signals of a block in box PLC_x (marked as B) corresponds to the arrangement of signal names in the header section (marked as A). Blocks B repeat in box PLC_x from top to bottom as a function of time.

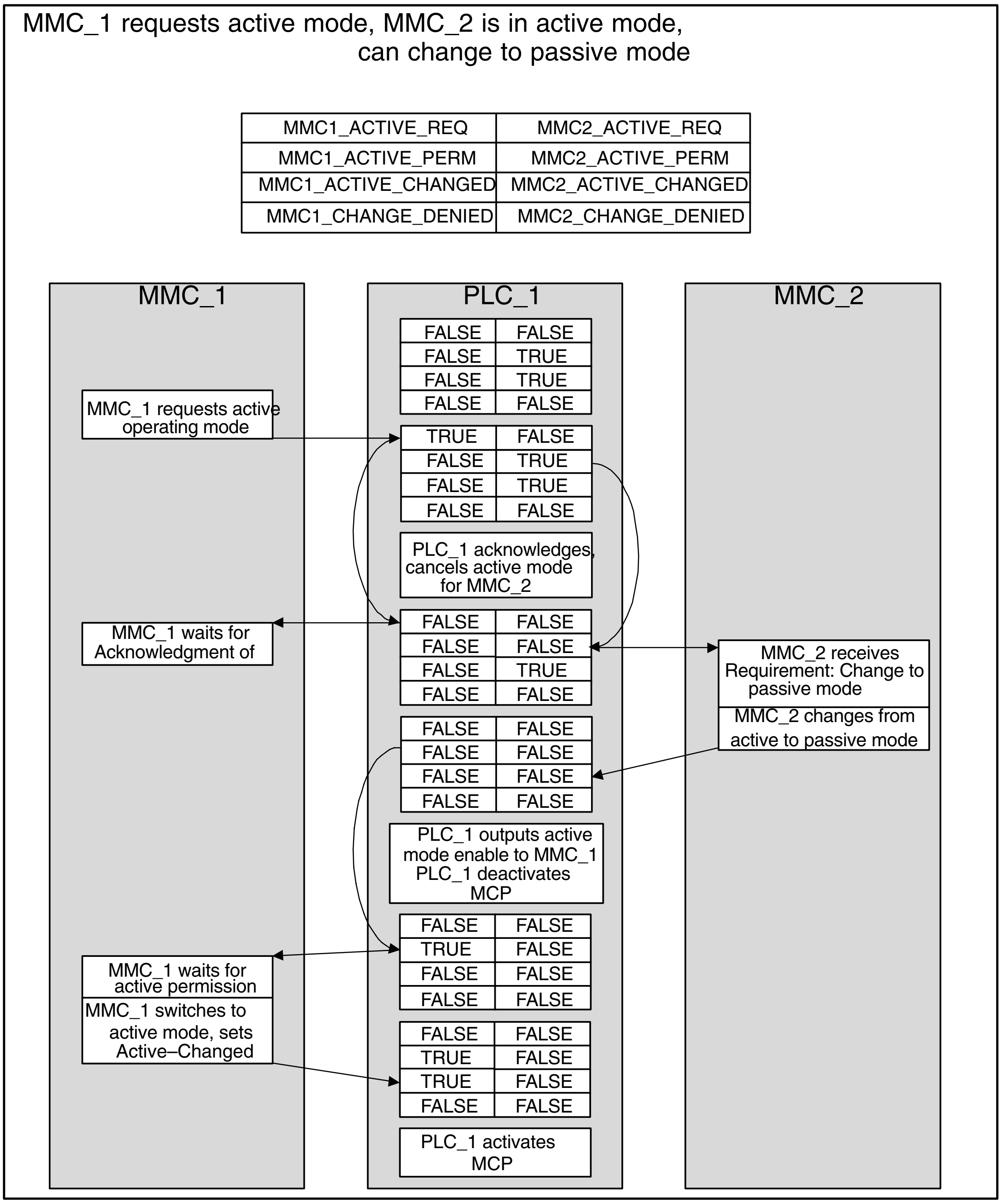



Fig. 6-2 MMC_1 requests active mode, MMC_2 is in active mode, can change to passive mode

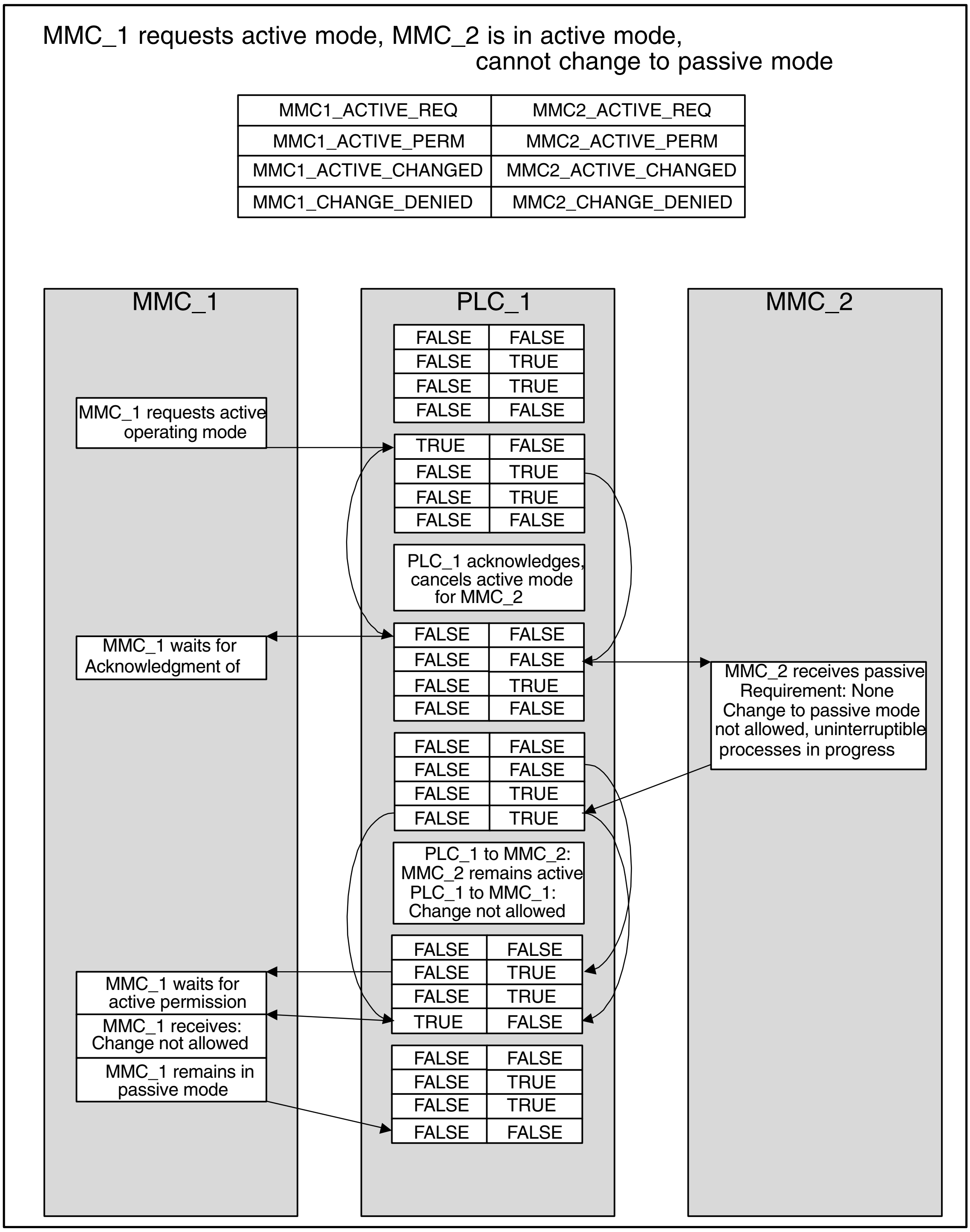



Fig. 6-3 MMC_1 requests active mode, MMC_2 is in active mode, but cannot change to passive mode

**MCP SWITCHOVER**

A control unit consists of an MMC and an MCP; these can both be switched over as a unit.

If an MCP has been configured for the MMC in configuring file NETNAMES.INI, it will be activated and deactivated with the MMC.

The MCP of whichever MMC is currently in active operating mode is activated.

In other words, only **one** MCP is ever active at any time on an NCU.

The MCP is activated by the PLC:

– MMC changes to active operating mode. (signal MMCx_ACTIVE_CHANGED: FALSE –> TRUE, x = 1.2 first or second MMC-PLC interface)

The MCP is deactivated by the PLC

– MMC changes to passive operating mode (signal MMCx_ACTIVE_CHANGED: TRUE –> FALSE, x = 1.2 first or second MMC-PLC interface)
– MMC goes offline as a result of switchover or displacement
The MMC sign-of-life signal changes from TRUE to FALSE when an MMC goes offline. After the edge change, the PLC deactivates the allocated MCP.
– Server MMC disconnects operating focus from the current NCU and switches it over to another. The server transmits S_ACT/ DIS_FOCUS as the last signal on its own MMC-PLC interface. The PLC then deactivates the corresponding MCP.

### 6.2.3 Defined logical functions/defines

---

**Note**

Please refer to Section 5.1 for the legal values for bus type, functions/status and additional information plus permissible combinations of status and additional information. The logical identifiers of functions are used in the following diagrams.

---

## 6.2.4 Graphic representation of function sequences

**Overview**

Figs. 6-4 to 6-9 describe the switchover operation for an operator station and Figs. 6-10 to 6-12 the switchover operation for a server.

The diagrams describe how an operator station is switched over (switchover from NCU_1 to NCU_2).

If an MMC in the offline status wishes to go online to an NCU (e.g.: during power-up), sequence OFFL_REQ_OP (...) and OFFL_CONF_OP(...) is omitted.

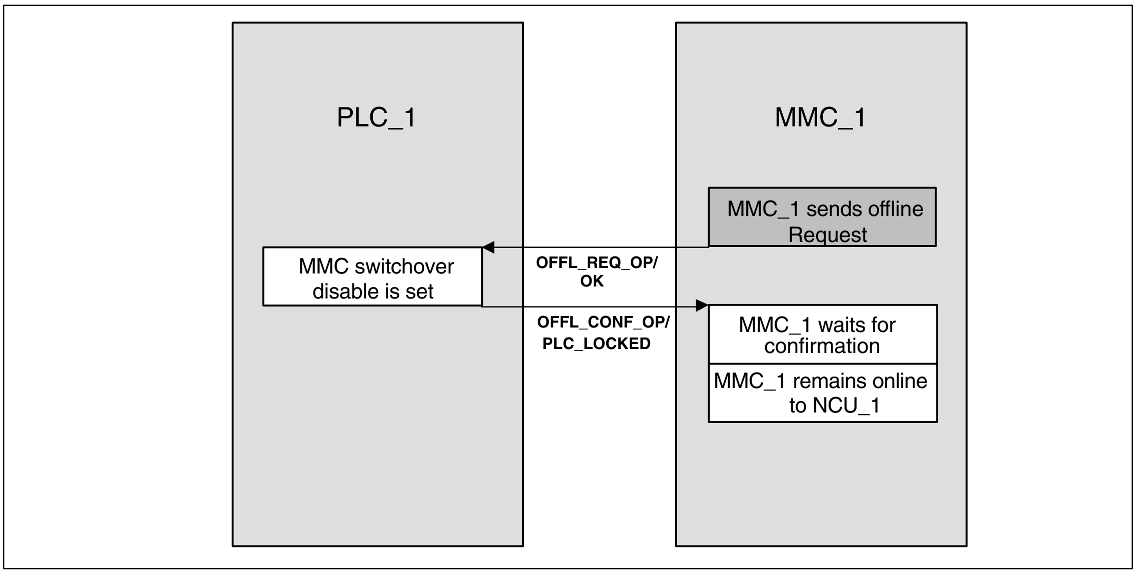



Fig. 6-4 MMC_1 is linked online to NCU_1 and wants to switch over to NCU_2, switchover disable is set in PLC_1

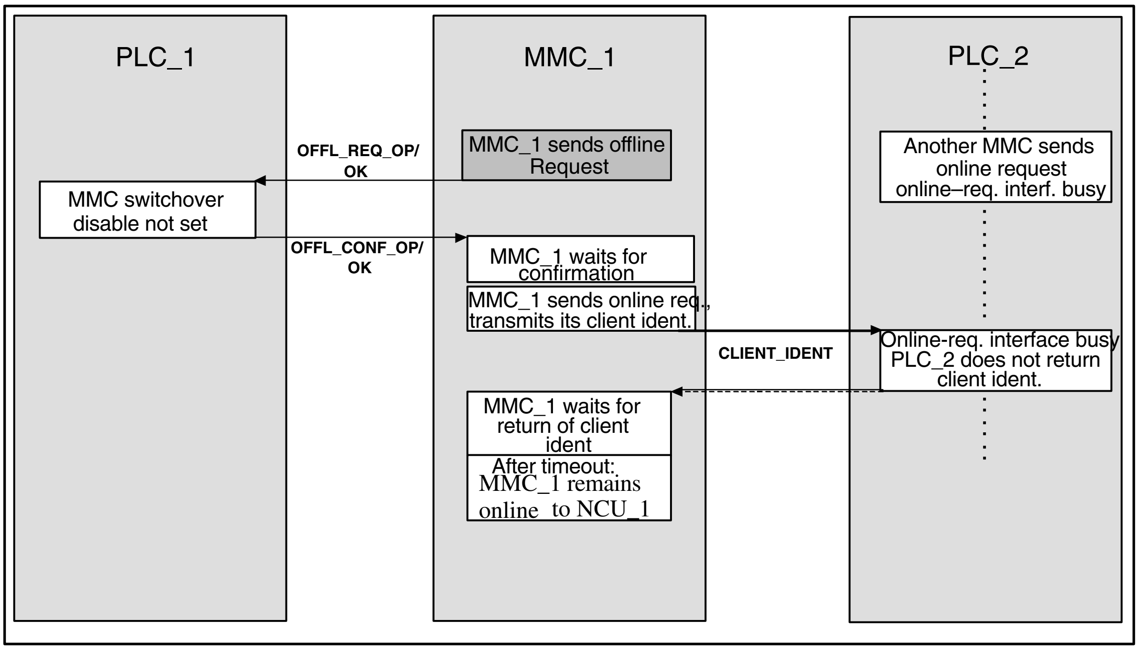



Fig. 6-5 MMC_1 online to NCU_1, MMC_1 wants to switch over to NCU_2, online-request interface in PLC_2 occupied by another MMC

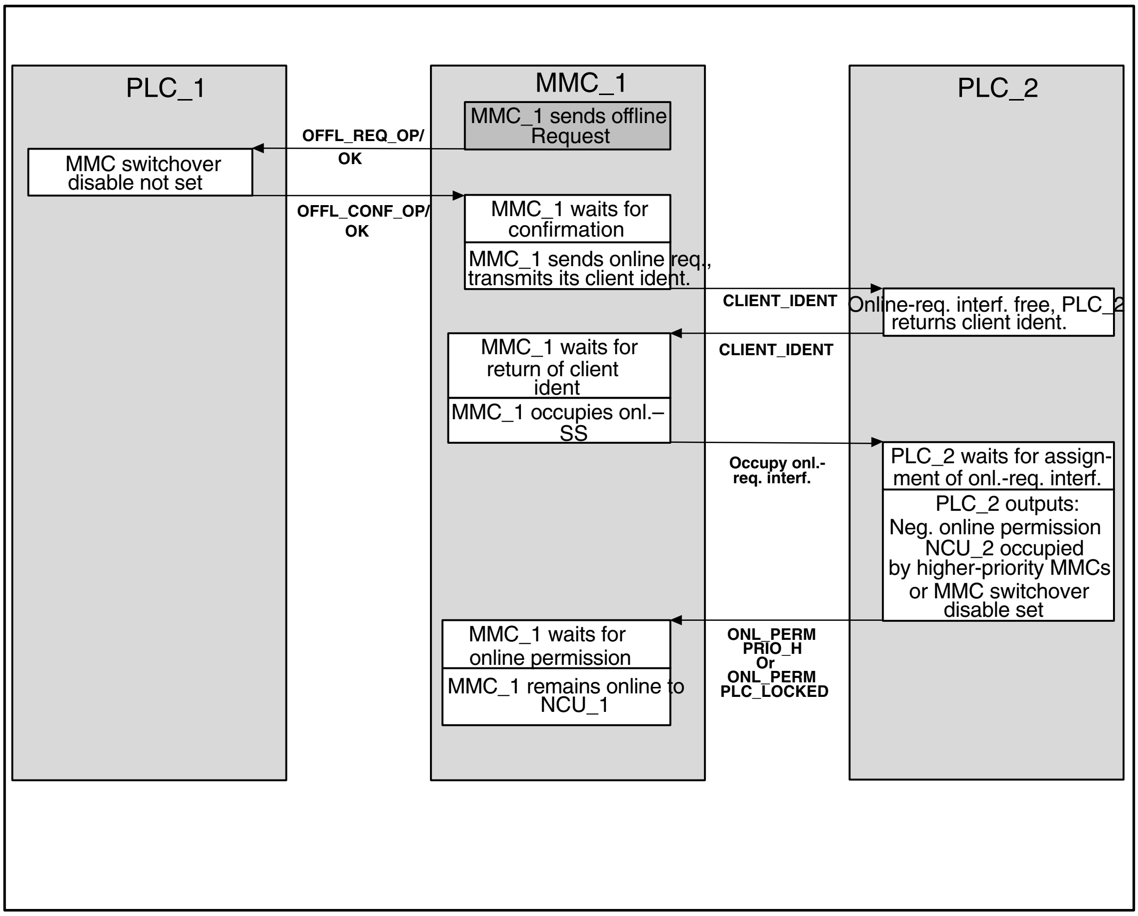



Fig. 6-6 MMC_1 online to NCU_1, MMC_1 wants to switch over to NCU_2, but does not receive permission from PLC_2

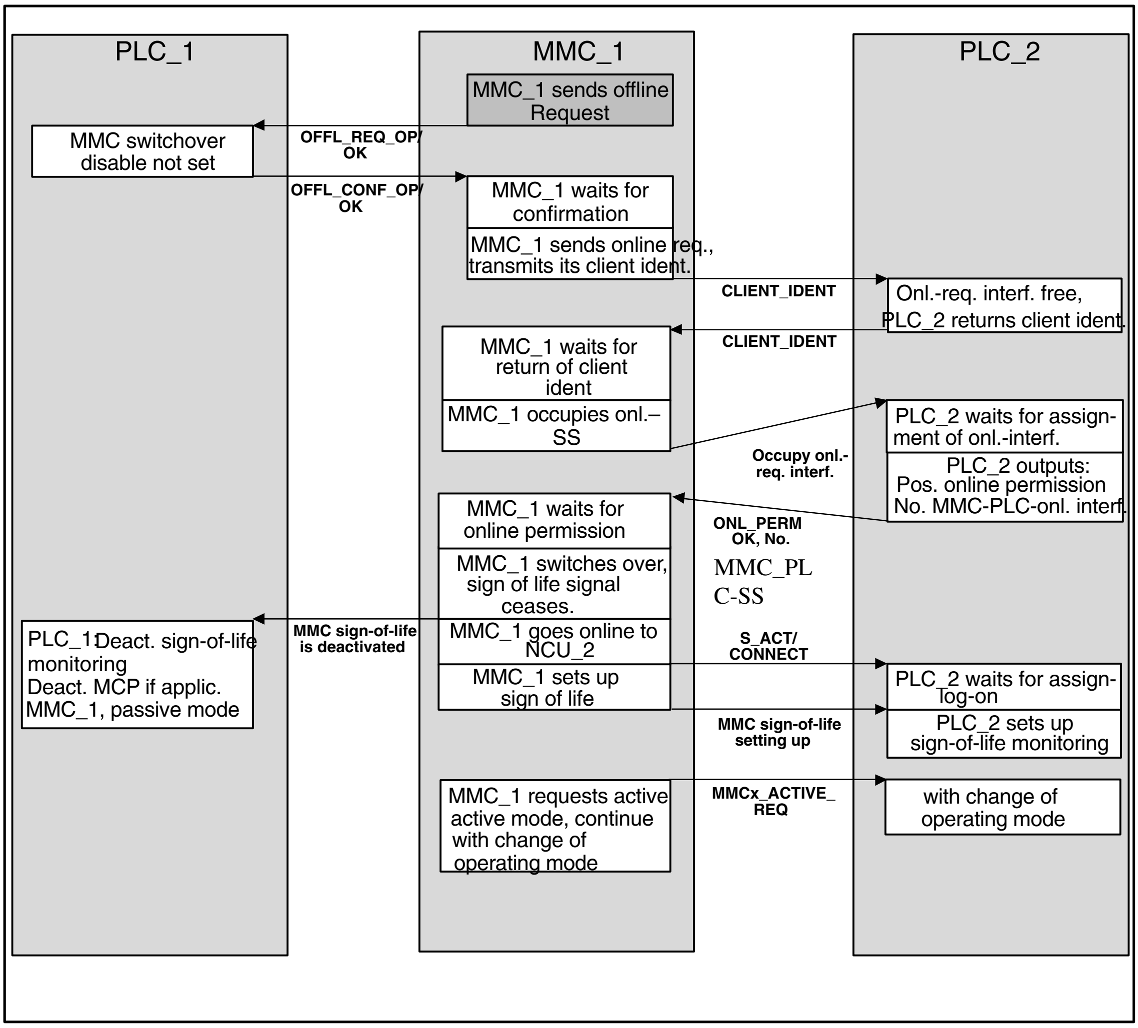



Fig. 6-7 MMC_1 online to NCU_1, MMC_1 switches over to NCU_2 (no displacement)

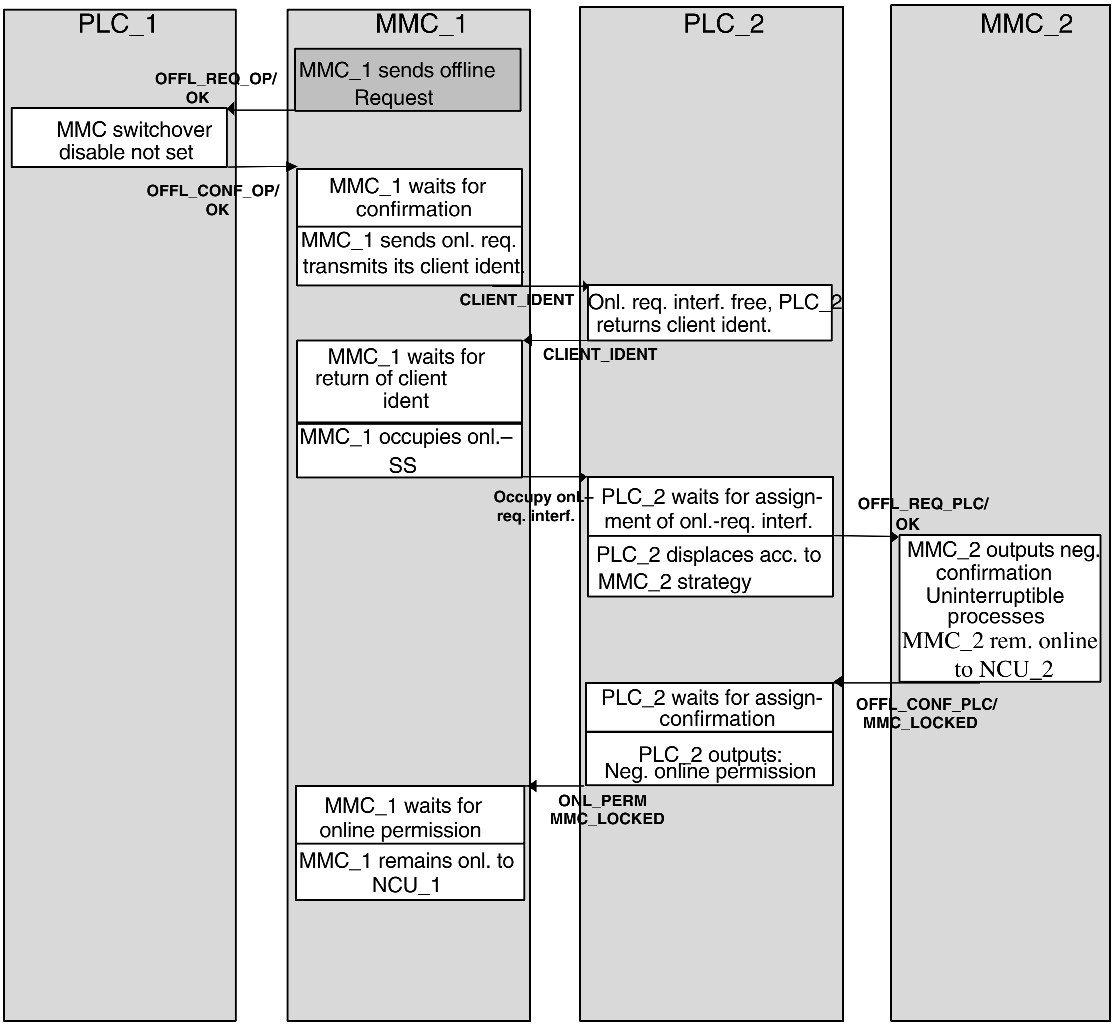



Fig. 6-8 MMC_1 online to NCU_1, MMC_2 online to NCU_2, MMC_1 wants to switch over to NCU_2, but MMCs executing uninterruptible processes are online to NCU_2

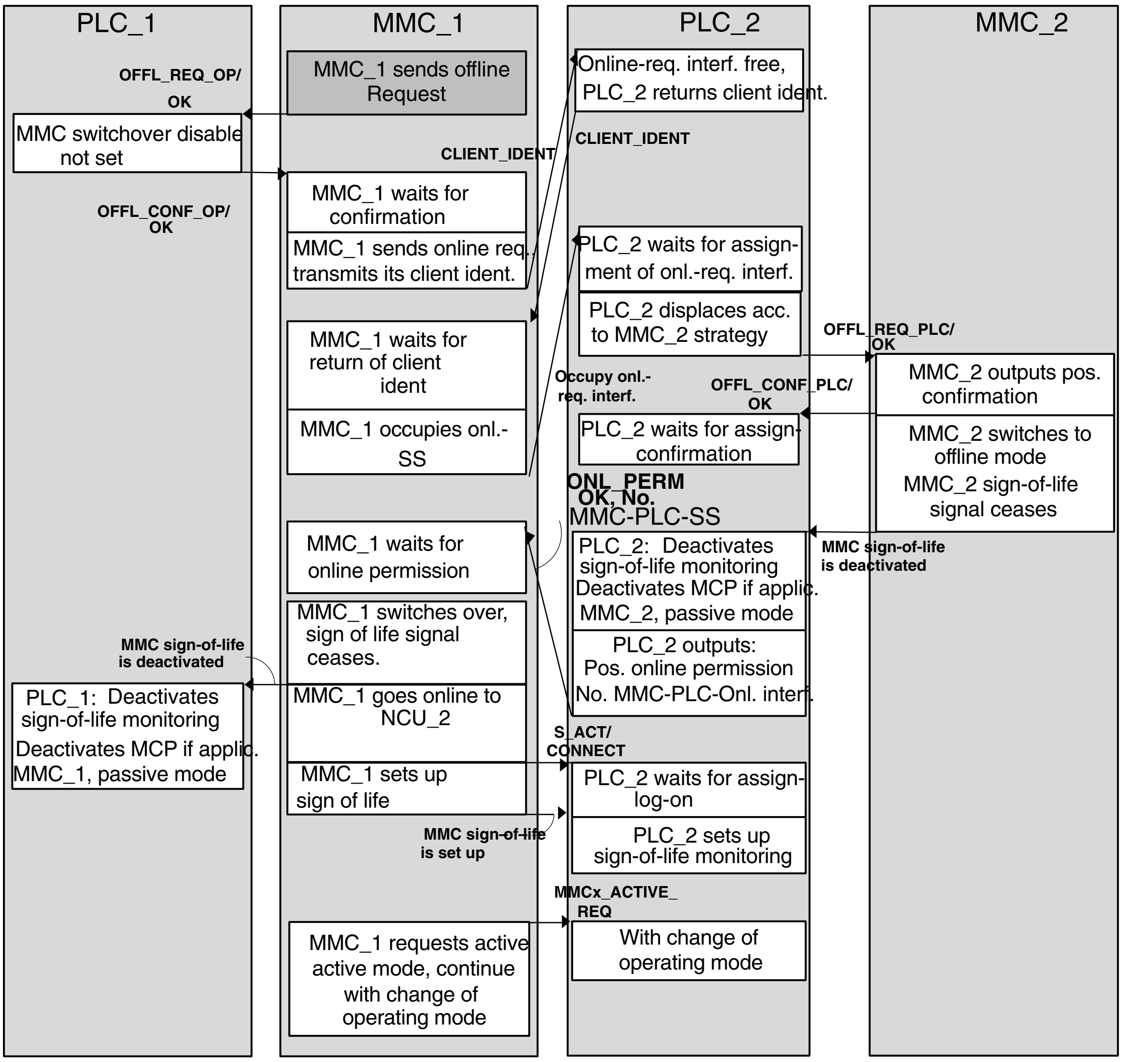



Fig. 6-9 MMC_1 online to NCU_1, MMC_2 online to NCU_2, MMC_1 switches from NCU_1 to NCU_2, MMC_2 is displaced

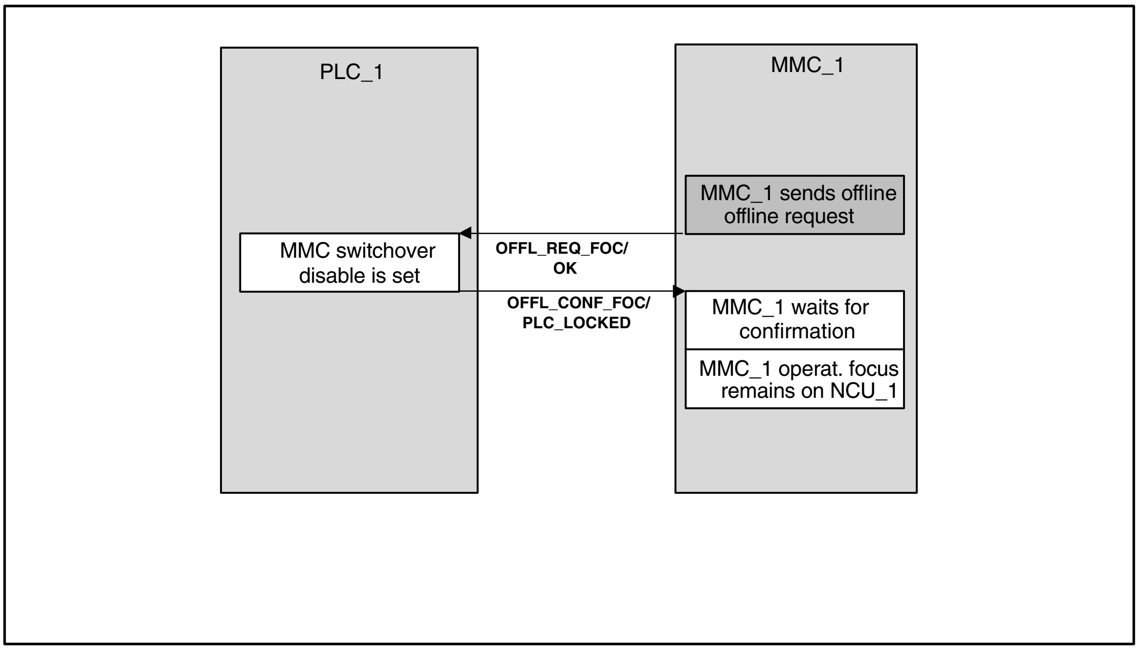



Fig. 6-10 MMC_1 server, wishes to switch operating focus from NCU_1 to NCU_2, switchover disabled in PLC_1

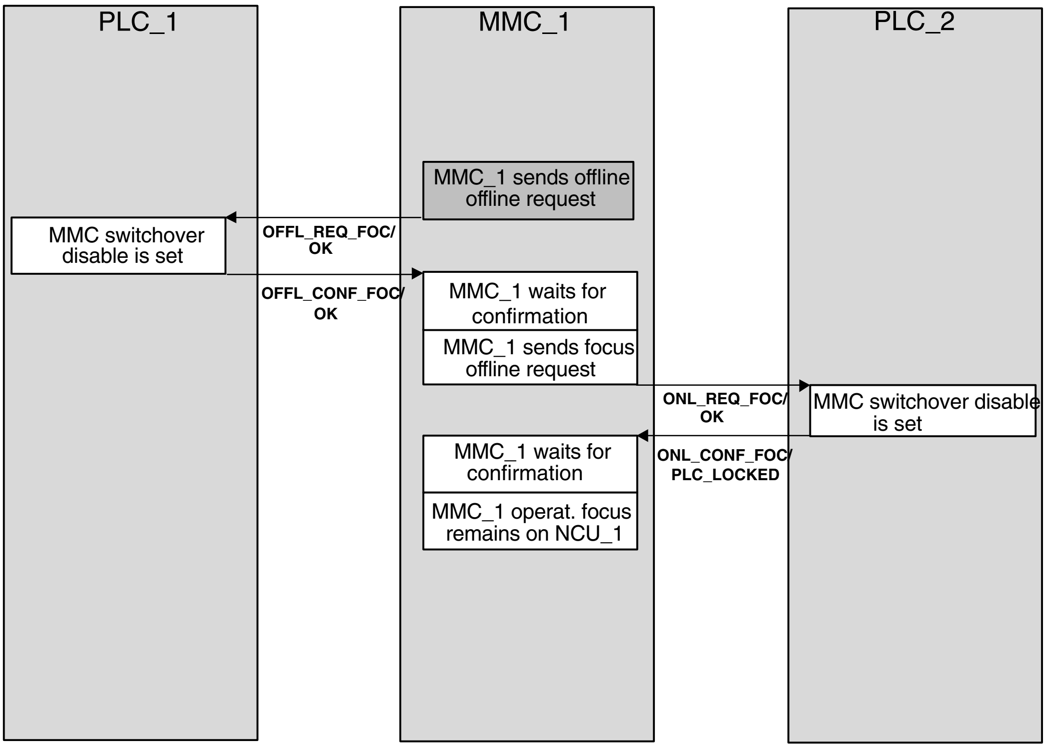



Fig. 6-11 MMC_1 is server, wishes to switch operating focus from NCU_1 over to NCU_2, switchover is disabled in PLC_2

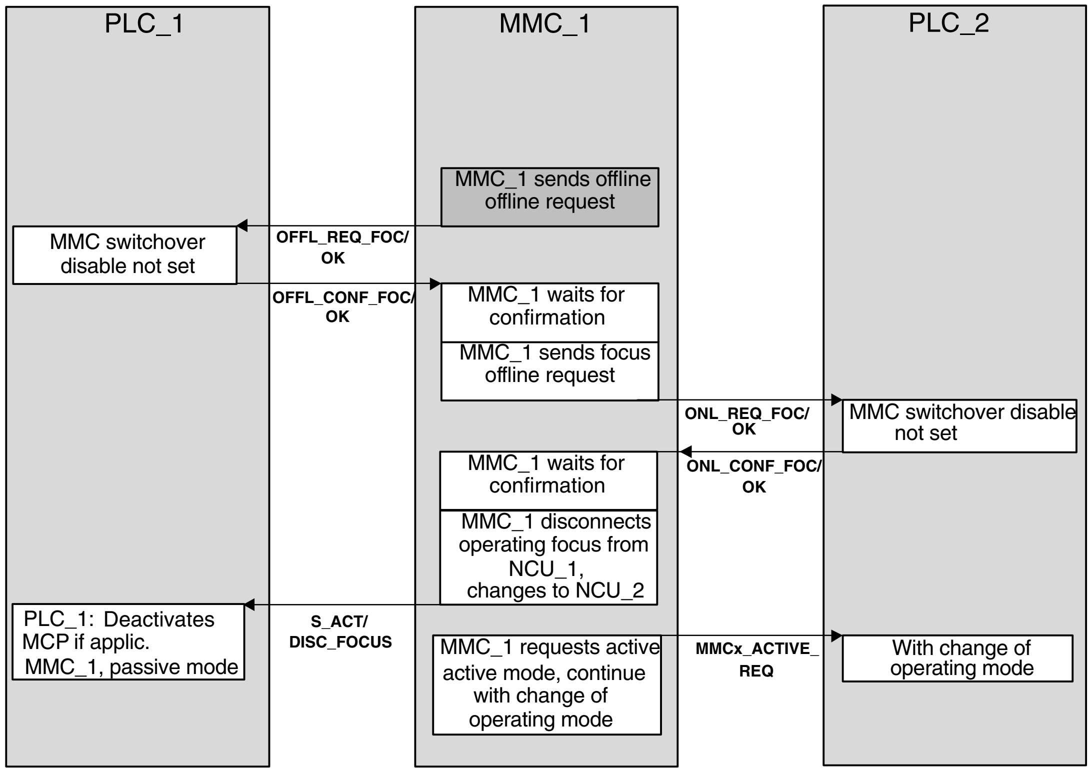



Fig. 6-12 MMC_1 is server, wishes to switch operating focus from NCU_1 over to NCU_2, switchover not disabled in PLCs, MMC_1 can change operating focus

# 6.3 Configuration file NETNAMES.INI, standard functionality

## 6.3.1 Two operator panel fronts and one NCU

For a system according to SW 3.1 (consisting of two control units and one NCU on the OPI, see Chapter 1), a sample configuration file for the second control unit is detailed below.
For explanations, see Chapter 1 “Configurability”.

```
; NETNAMES.INI Example 2 Start

; Identification entry
[own]
owner =         MMC_2

; Connection entry
[conn MMC_1]
conn_1          NCU_1

[conn MMC_2]
conn_1          NCU_1

; Descriptive entry
[param network]
bus =           btss

[param MMC_1]
mmc_address =   1

[param MMC_2]
mmc_address =   3

[param NCU_1]
nck_address =   13
plc_address =   13

; NETNAMES.INI Example 2 end
```

### 6.3.2 One operator panel front and three NCUs

For a system according to SW 3.2 (consisting of one control unit and three NCUs on the OPI, see Chapter 1, paragraph “Configurability”), a sample configuration file is detailed below.
For explanations, see Chapter 1, paragraph “Configurability”.

Any adaptations which may need to be made are described in Chapter 2, paragraph “Configurations”.

```
; NETNAMES.INI Example 3 Start
; Identification entry
[own]
owner =         MMC_1

; Connection entry: For the total of 3 connections provided
[conn MMC_1]
conn_1=         NCU_1
conn_2=         NCU_2
conn_3=         NCU_3

; Descriptive entry: The network is clearly defined
[param network]
bus=            opi

[param MMC_1]
name=           arbitrary_name
type=           MMC_100
mmc_address=    1

[param NCU_1]
name=           arbitrary_name1
type=           ncu_572
nck_address=    12
plc_address=    12

[param NCU_2]
name=           arbitrary_name2
type=           ncu_573
nck_address=    14
plc_address=    14

[param NCU_3]
name=           arbitrary_name3
type=           ncu_573
nck_address=    15
plc_address=    15

; NETNAMES.INI, example 3 end
```

## 6.4 M:N quick installation guide with examples

**Introduction**

The MPI/OPI bus network rules are not described.

See /BH/, Operator Components Manual

Three examples are used to demonstrate the steps involved in starting up an M:N interconnection. Each description begins by presenting a configuration. If you do not find your configuration here, we recommend you use the detailed description in Chapter 2 and make the appropriate additions or amendments.

### 6.4.1 Example 1

**Hardware configuration**

The hardware comprises the following components:

– 1 MMC103 / PCU50

– 1 MCPs

– Two NCUs with two channels each

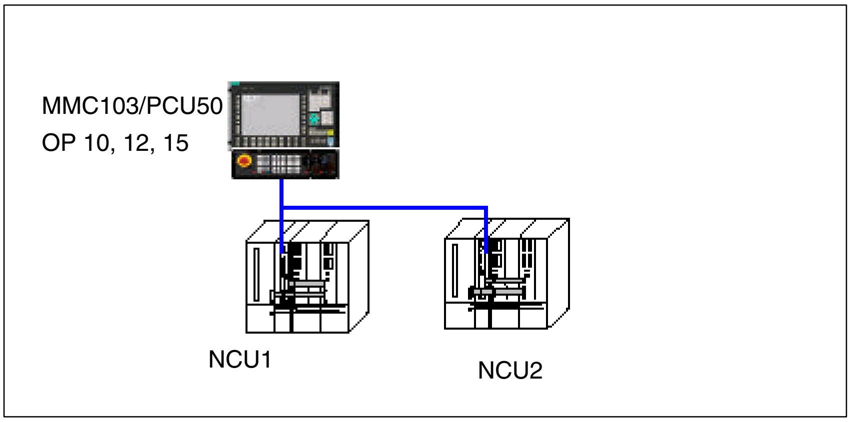



Fig. 6-13 One operator panel for two NCUs

**Use**

Operator panel 1 (MMC103/PCU50 server) with machine control panel can access NCU1 (channel 1, channel 2) and NCU2 (channel 1, channel 2).

**Step 1:**

**Configuration file NETNAMES.INI**

For this example, the following entries are set in the NETNAMES.INI file:

[own]
owner= MMC_1

```
; Connection entry
[conn MMC_1]
conn_1 = NCU_1
conn_2 = NCU_2
; Extcall not required for a PCU
; Network parameters
[param network]
bus = opi
; MMC definitions
[param MMC_1]
mmc_address = 1
; All other parameters not required
; NCU components descriptive entry
[param NCU_1]
type = NCU_573
nck_address = 22
plc_address = 22
name = NCU1

[param NCU_2]
type = NCU_573
nck_address = 23
plc_address = 23
name = NCU2

; Channel data
[chan MMC_1]
DEFAULT_logChanSet = Station_1
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_1, Station_2

[Station_1]
logChanList = N1_K1, N1_K2
[N1_K1]
logNCName = NCU_1
ChanNum = 1
[N1_K2]
logNCName = NCU_1
ChanNum = 2
```

[Station_2]
logChanList = N2_K1, N2_K2

[N2_K1]
logNCName = NCU_2
ChanNum = 1

[N2_K2]
logNCName = NCU_2
ChanNum = 2

; End

**Step 2:**

**Load file NETNAMES.INI**

**MMC103/PCU50:** Once the NETNAMES.INI file has been created, it is transferred into the USER directory of the operator panel.

**Step 3:**

**Set the NCK bus addresses**

Starting at the main screen of the operator interface, the action:
Start-up → MMC → operator panel front opens the “Operator panel front interface parameters” input window. Enter the following here:

| | | |
|---|---|---|
| Connection: | M:N | Select M:N instead of 1:1 |
| NCK address: | 22 | |
| PLC address: | 22 | |

; according to NETNAMES.INI also for NCU2 address 23

- Select “Save”
- Restart the operator panel

**Step 4:**

**PLC**

An FB9 call is not required for this configuration, because no displacement or active/passive switching takes place.

**Softkey designation**

In order to distinguish which NCU is to be addressed, texts must be defined for the inscription of the softkeys of the operator panel for MMC100.2/PCU20. See example 2. For MMC103/PCU50, the texts are copied from the NETNAMES.INI file. No extra texts over and above those in NETNAMES.INI are required for the present example.

### 6.4.2 Example 2

**Hardware configuration**

The hardware comprises the following components:

- 1 MMC103 / PCU50
- 1 MMC100.2 / PCU20
- 2 MCPs
- Two NCUs with two channels each

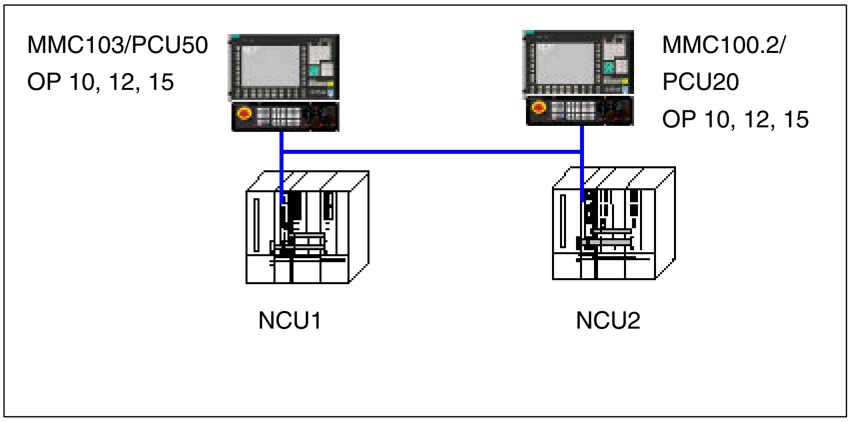



Fig. 6-14 Two operator panels for two NCUs

**Use**

Control panel 1 (server) can access NCU1 (channel 1, channel 2) and NCU2 (channel 1, channel 2) with machine control panel switchover.
Control panel 2 (secondary control panel) can also access NCU1 and NCU2.

**Step 1a):**

**NETNAMES.INI configuration files**

In this example, own entries are input for the operator panels in NETNAMES.INI files.

**Operator panel 1**

Entries for MMC 103 / PCU 50:

[own]
owner= MMC_1

**; Connection entry**

[conn MMC_1]
conn_1 = NCU_1
conn_2 = NCU_2

EXTCALL_conns = conn_1, conn_2

```
; Network parameters

[param network]
bus = opi

; MMC definitions

[param MMC_1]
mmc_typ = 0x40
mmc_bustyp = OPI
mmc_address = 1
mstt_address = 6
name = MMC_Serv
start_mode = ONLINE

; NCU components descriptive entry

[param NCU_1]
type = NCU_573
nck_address = 22
plc_address = 22
name = NCU1


[param NCU_2]
type = NCU_573
nck_address = 23
plc_address = 23
name = NCU2


; Channel data

[chan MMC_1]
DEFAULT_logChanSet = Station_1
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_1, Station_2


[Station_1]
logChanList = N1_K1, N1_K2

[N1_K1]

logNCName = NCU_1
ChanNum = 1

[N1_K2]
logNCName = NCU_1
ChanNum = 2


[Station_2]
logChanList = N1_K1, N1_K2

[N1_K1]
logNCName = NCU_2
ChanNum = 1

[N1_K2]
logNCName = NCU_2
ChanNum = 2

; End
```

**Step 2a):**

**Load file NETNAMES.INI**

**MMC103/PCU50:** After the NETNAMES.INI is created, it is transferred to the USER directory of the MMC 103/PCU 50 control unit.

**Step 1b):**

**Operator panel 2**

Entries for MMC 100.2/PCU 20:

[own]
owner= PCU 20

**; Connection entry**

[conn PCU20]
conn_1 = NCU_1
conn_2 = NCU_2

**; Network parameters**

[param network]
bus = opi

**; MMC definitions**

[param PCU20]
mmc_typ = 0x10
mmc_bustyp = OPI
mmc_address = 2
mstt_address = 7
name = MMC_Neben
start_mode = OFFLINE

**; NCU components descriptive entry**

[param NCU_1]
type = NCU_573
nck_address = 22
plc_address = 22
name = NCU1

[param NCU_2]
type = NCU_573
nck_address = 23
plc_address = 23
name = NCU2

**; Channel data**

[chan PCU20]
DEFAULT_logChanSet = Station_2
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_1, Station_1

[Station_1]
logChanList = N1_K1, N1_K2

[N1_K1]
logNCName = NCU_1
ChanNum = 1

[N1_K2]
logNCName = NCU_1
ChanNum = 2

[Station_2]
logChanList = N1_K1, N1_K2

[N1_K1]
logNCName = NCU_2
ChanNum = 1

[N1_K2]
logNCName = NCU_2
ChanNum = 2

; End

**Softkey designation**

In order to distinguish which NCU is to be addressed, texts must be defined in **chan.txt** for the inscription of the soft keys of the operator panel:

//* Max. length of text 2*9 characters */

//* Create new line with **%n** at the end of the first line */

//* Name of channel area 1 and names of the channels of this area */

T_CHAN_AREA_1 “Stat_1”
T_CHAN_AREA_1_CHANNEL_1 “N1_K1”
T_CHAN_AREA_1_CHANNEL_2 “N1_K2”

//* Name of channel area 2 and names of the channels of this area */

T_CHAN_AREA_2 “Stat_2”
T_CHAN_AREA_2_CHANNEL_1 “N2_K1”
T_CHAN_AREA_2_CHANNEL_2 “N2_K2”

**Step 2b:**

**PCU20**

After the NETNAMES.INI and chan.txt files have been created, they are included in the *.abb file with the application.

**Step 3:**

**Set the NCK bus addresses**

**MMC103/PCU50:**
Starting at the main screen of the operator interface, the action:
Start-up → MMC → operator panel front opens the “Operator panel front interface parameters” input window. Enter the following here:

| | | |
|---|---|---|
| Connection: | M:N | instead of 1:1 |
| NCK address: | 22 | |
| PLC address: | 22 | |

; according to NETNAMES.INI also for NCU2 address 23

- Select “Save”
- Restart the operator panel

**MMC100.2/PCU20:**
Transfer *.abb onto the system using a PC card and perform a software update.

---

**Note**

If you have forgotten to include the “chan.txt” file in *.abb, no inscribed soft keys are visible when you select the channel menu key. The selection function is available, however.

---

**Step 4:**

**PLC**

Include FB9 in the PLC user program. You will find more details after the examples below.

### 6.4.3 Example 3

**Hardware configuration**

The hardware comprises the following components:

– MMC103/PCU50
– 1 HT6
– 1 MCP
– 2 NCUs with two channels each

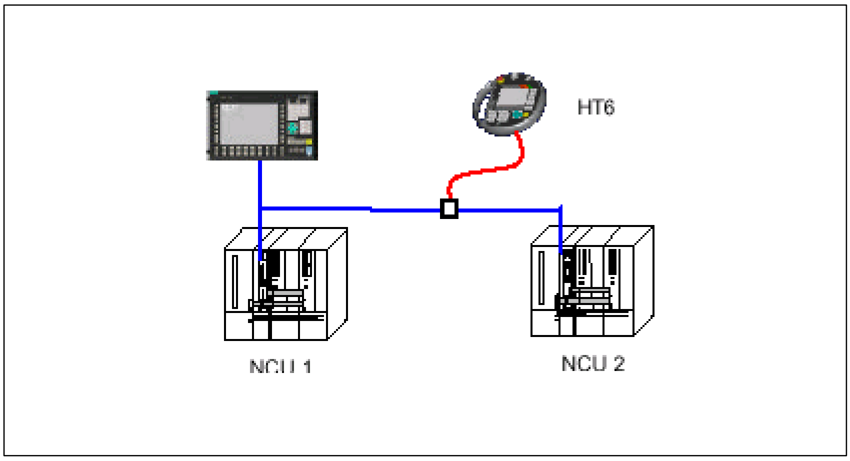



Fig. 6-15 Operator panel front and HT6 for two NCUs

**Use**

Operator panel 1 (MMC 103/PCU 50 server) without machine control panel can access NCU1 (channel 1, channel 2) and NCU2 (channel 1, channel 2). Operator panel 2 (secondary control panel, HT6) can only access NCU2, HT6 is implicitly the MCP.

**Step 1a:**

Create the NETNAMES.INI file for MMC 103/PCU 50

[own]
owner= MMC_1

**; Connection entry**

[conn MMC_1]
conn_1 = NCU_1
conn_2 = NCU_2

EXTCALL_conns = conn_1, conn_2

**; Network parameters**

[param network]
bus = opi

```
; MMC definitions

[param MMC_1]
mmc_typ = 0x40
mmc_bustyp = OPI
mmc_address = 1
mstt_address = 255            ; 255 is required if no MCP
                              ; is installed.
name = MMC_Serv
start_mode = ONLINE

; Description of NCU components

[param NCU_1]
type = NCU_573
nck_address = 22
plc_address = 22
name = NCU1


[param NCU_2]
type = NCU_573
nck_address = 23
plc_address = 23
name = NCU2

; Channel data

[chan MMC_1]
DEFAULT_logChanSet = Station_1
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_1, Station_2

[Station_1]
logChanList = N1_K1, N1_K2

[N1_K1]
logNCName = NCU_1
ChanNum = 1

[N1_K2]
logNCName = NCU_1
ChanNum = 2


[Station_2]
logChanList = N1_K1, N1_K2

[N1_K1]
logNCName = NCU_2
ChanNum = 1

[N1_K2]

logNCName = NCU_2
ChanNum = 2

; End
```

**Step 1b:**

Create the NETNAMES.INI file for operator panel 2 (HT6)

[own]
owner = HT_6

**; Connection entry**

[conn HT_6]
conn_1 = NCU_2

**; Network parameters**

[param network]
bus = opi

**; MMC definitions**

[param HT_6]
mmc_typ = 0x10
mmc_bustyp = OPI
mmc_address = 14
mstt_address = 14 ; is always identical to
; mmc address
name = MMC_Neben
start_mode = OFFLINE

**; Description of NCU components**

[param NCU_2]
type = NCU_573
nck_address = 23
plc_address = 23
name = NCU2

**; Channel data**

[chan HT_6]
DEFAULT_logChanSet = Station_2
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_2

[Station_2]
logChanList = N2_K1, N2_K2

[N2_K1]
logNCName = NCU_2
ChanNum = 1

[N2_K2]
logNCName = NCU_2
ChanNum = 2

;End of file

**Step 2a:**

**MMC/PCU50:**
After the NETNAMES.INI is created, it is transferred or copied to the USER directory of the corresponding MMC/PCU.

**Step 2b:**

**HT6:**

See example 2 for the creation of the soft key texts.

After the NETNAMES.INI and chan.txt files have been created, they are included in the *.abb file with the application.

**Step 3:**

**Set the NCK bus addresses**

**MMC103/PCU50:**
Starting at the main screen of the operator interface, the action:
*Start-up* → *MMC* → *operator panel front* opens the "Operator panel front interface parameters" input window. Enter the following here:

| | | |
|---|---|---|
| Connection: | M:N | instead of 1:1 |
| NCK address: | 22 | |
| PLC address: | 22 | |

; according to NETNAMES.INI also for NCU2 address 23

- Select "Save"
- Restart the operator panel

**Step 4:**

Include FB9 in the PLC user program. You will find more details in the following section.

### 6.4.4 Description of FB9

**Description of functions**

This block allows **switchover** between several **control units** (MMC operator panel fronts and/or MCP machine control panels) which are connected to one or more NCU control modules over a bus system.

The **interface** between the individual control units and the NCU (PLC) is the M : N interface in **data block DB19** (see Chapter 5 Signal Descriptions and /LIS/, Lists, Chapter 4).
FB 9 uses the signals of this interface.

Apart from initialization, sign-of-life monitoring and error routines, the following **basic functions** are also performed by the block for control unit switchover:

Table 6-3 Overview of functions

| Basic function | Meaning |
|---|---|
| MMC call waiting | MMC wants to go online to an NCU |
| MMC coming | MMC is connecting to an NCU |
| MMC going | MMC is disconnecting from an NCU |
| Forced break | MMC must break connection to an NCU |
| Operating focus changeover to server mode | Change operating focus from one NCU to the other |
| Active/passive operating mode | Operator control and monitoring/monitoring only |
| MCP switchover | As an option, MCP can be switched over with the MMC |

The following descriptions supplement the information in 6.2.1 and 6.2.2 with particular reference to the behavior in the last three examples.

**Brief description of important functions**

**Active/passive operating mode**

An online MMC can operate in two different modes:

Active mode: Operator can control and monitor

Passive mode: Operator can monitor (MMC header only)

After switchover to an NCU, this initially requests active operating mode in the PLC of the online NCU. If two MMCs are simultaneously connected online to one NCU, one of the two is always in active and the other in passive operating mode. The operator can request active mode on the passive MMC at the press of a button.

**MCP switchover:**

As an option, an MCP assigned to the MMC can be switched over at the same time. To achieve this, the MCP address must be entered in the **mstt_adress** parameter of the NETNAMES.INI configuration file of the MMCs and **MCPEnable** set to true. The MCP of the passive MMC is deactivated. This means only one MCP is ever active on an NCU.

**Power-up condition:**

To prevent the previously selected MCP from being activated again when the NCU is restarted, input parameters **MCP1BusAdr = 255** (address of 1st MCP) and **MCP1STOP =TRUE** (deactivate 1st MCP) must be set when FB1 is called in OB100.

**Enabling commands:**

When one MCP is switched over to another, any active feedrate or axis enabling signals may be transferred at the same time.

**Important:**

Keys actuated at the moment of switchover remain operative until the new MCP is activated (by the MMC which is subsequently activated). The override settings for feedrate and spindle also remain valid. To deactivate actuated keys, the input image of the machine control signals must be switched to nonactuated signal level on a falling edge of DB10.DBX104.0. The override settings should remain unchanged.

Measures for deactivating keys must be implemented in the PLC user program. (see below: Example of override switchover)

### Declaration of the function

**FUNCTION_BLOCK FB9**

```
VAR_INPUT
Quit : BOOL;                       // Acknowledge alarms
OPMixedMode : BOOL:= FALSE ;       // Hybrid operation with non M:N-capable
                                   // OP deactivated!
AktivEnable : BOOL:= TRUE ;         // Activate active/passive switchover.
MCPEnable : BOOL:= TRUE ;           // Activate MCP switchover
END_VAR

VAR_OUTPUT
Alarm1 : BOOL ;                    // Alarm: Error in MMC bus address, bus
type!
Alarm2 : BOOL ;                    // Alarm: No confirmation MMC1 offline!
Alarm3 : BOOL ;                    // Alarm: MMC1 is not going offline!
Alarm4 : BOOL ;                    // Alarm: No confirmation MMC2 offline!
Alarm5 : BOOL ;                    // Alarm: MMC2 is not going offline!
Alarm6 : BOOL ;                    // Alarm: Queuing MMC is not going online!
Report : BOOL ;                    // Alarm: Sign-of-life monitoring
ErrorMMC : BOOL ;                  // Error detection MMC
END_VAR
```

**Explanation of the formal parameters**

The following table shows all formal parameters of function FB9

Table 6-4 Formal parameters of FB9

| Signal | Type | Type | Remarks |
|---|---|---|---|
| Quit | E | BOOL | Acknowledge alarms |
| OPMixedMode | E | BOOL | Hybrid operation with non M:N-capable OP |
| AktivEnable | E | BOOL | Activate active/passive operator panel switchover<br>TRUE = Operator panel can be switched to active/passive.<br>FALSE = Operator panel cannot be switched to active/passive and remains in its current state. |
| MCPEnable | E | BOOL | Activate MCP switchover<br>TRUE = MCP is switched with operator panel.<br>FALSE: = MCP is not switched with operator panel. |
| Alarm1 | A | BOOL | Alarm: Error in MMC bus address, bus type! |
| Alarm2 | A | BOOL | Alarm: No confirmation MMC1 offline! |
| Alarm3 | A | BOOL | Alarm: MMC1 is not going offline! |
| Alarm4 | A | BOOL | Alarm: No confirmation MMC2 offline! |
| Alarm5 | A | BOOL | Alarm: MMC2 is not going offline! |
| Alarm6 | A | BOOL | Alarm: Queuing MMC is not going online! |
| Report | A | BOOL | Message: Signoflife monitoring |
| ErrorMMC | A | BOOL | Error detection MMC |

**Note**

The block must be called by the user program. The user must provide an instance DB with any number for this purpose. The call is not multi-instance-capable.

### 6.4.5 FB9 call

```
CALL FB 9 , DB 109 (

Quit := Fehler_Quitt,              // e.g. MCP reset
OPMixedMode := FALSE,
AktivEnable := TRUE,               // Enable MMC switchover
MCPEnable := TRUE,                 // Enable MCP switchover
Alarm1 := DB2.dbx188.0,            // Error message 700.100
Alarm2 := DB2.dbx188.1,            // Error message 700.101
Alarm3 := DB2.dbx188.2,            // Error message 700.102
Alarm4 := DB2.dbx188.3,            // Error message 700.103
Alarm5 := DB2.dbx188.4,            // Error message 700.104
Alarm6 := DB2.dbx188.5,            // Error message 700.105
Report := DB2.dbx192.0,            // Operational message 700.132
ErrorMMC := DB2.dbx192.1)          // Operational message 700.133
```

---

**Note**

AktivEnable := true enables MMC active/passive switchover.

MCPEnable := true allows MCP switchover.

The default value of this parameter is thus enabled and does not have to be enabled explicitly when the function is called.

---

**Alarms, errors**

The output parameters “Alarm1” to “Alarm6” and “Report” can be passed in the DB2 areas for MMC alarm and error messages.

If execution of an MMC function has failed (for which an appropriate error message cannot be displayed), status parameter ErrorMMC is set to ’logic 1’ (e.g. booting error when no connection is made).

**Example call for FB1**

(call in OB100):

```
CALL “RUN_UP” , “gp_par” (

MCPNum := 1,
MCP1In := P#E 0.0,
MCP1Out := P#A 0.0,
MCP1StatSend := P#A 8.0,
MCP1StatRec := P#A 12.0,
MCP1BusAdr := 255,                 // Address of 1st MCP
MCP1Timeout := S5T#700MS,
MCP1Cycl := S5T#200MS,
MCP1Stop := TRUE,                  // MCP disabled
NCCyclTimeout := S5T#200MS,
NCRunupTimeout := S5T#50S);
```

### 6.4.6 Example of override switchover

The example uses auxiliary flags M100.0, M100.1, M100.2, M100.3.

The positive edge of MCP1Ready must check for override and initiate measures for the activation of the MCP block.

This example applies to the feedrate override. The interface and input bytes must be exchanged for spindle override.

```
A   DB10.DBX  104.0;            // MCP1Ready
    FN   M   100.0;             // Edge trigger flag 1
    JCN  wei1;
    S    M   100.2;             // Set auxiliary flag 1
    R    M   100.3;             // Reset auxiliary flag 2

// Save override

    L DB21.DBB 4;               // Feedrate override interface
    T IB 28;                    // Buffer memory (free input
                                // or memory byte)

wei1:

   A M 100.2;                   // Switchover
   O DB10.DBX 104.0;            // MCP1Ready
   JCN wei2;
   A DB10.DBX 104.0;            // MCP1Ready
   FP M 100.1;                  // Edge trigger flag 2
   JC wei2;
   A M 100.2;                   // Switchover
   R M 100.2;                   // Reset auxiliary flag 1
   SPB wei2;
   A M 100.3;                   // Comparison made
   JC MCP;                      // Call MCP program

// Route saved override to interface of switched MCP
// until the override values match

    L EB28;                     // Redirect buffer memory
    T DB21.DBB 4;               // to override interface
    L EB 3;                     // Override input byte for feed
    <>i;                        // Match ?
    JC wei2;                    // no, exit
    S   M100.3;                 // yes, set auxiliary flag 2
```

```
// When override values match, call the MCP program again
MCP: CALL "MCP_IFM" (            // FC 19
   BAGNo := B#16#1,
   ChanNo := B#16#1,
   SpindleIFNo := B#16#0,
   FeedHold := M 101.0,
   SpindleHold := M 101.1);
wei2: NOP   0;
```

### 6.4.7 Switchover between MCP and HT6

```
CALL FCxx
   L DB7.DBB 27                  // Act. MCP
   L 6                           // Machine control panel
   ==I
   JC MSTT                       // Call FC 19
   L DB7.DBB 27                  // Act. MCP
   L 14                          // HT 6
   ==I
   JC HT6                        // Call FC 26
   JU ENDE
HT6: NOP 0
     L B#16#40    // Shift inputs of HT6 to input byte 8+n
     T DB7.DBB7
     L B#16#40    // Shift outputs of HT6 to output byte 8+n
     T DB7.DBB13
     CALL FC26                   // Call HT6 block
     JU ENDE
MCP: NOP 0
     L 0
     T DB7.DBB7
     T DB7.DBB13

     CALL FC19                   // Call machine control panel block
END: NOP 0
```

### 6.4.8 General notes

- In a configuration with only **one** NCU, the additional entry : " ,SAP=202 " must be set for the PLC address in the [param NCU_xx] section of the NETNAMES.INI file.

Example:

[param NCU_1]
type =NCU_573
nck_address = 11
plc_address = 11**, SAP = 202**
name = NCU1

- In a configuration without a machine control panel (operator panel without MCP), "mstt_address = **255**" must be set in the [param MMC_xx] section of the relevant NETNAMES.INI file.
  This does not apply in the case of an MMC100.2 / PCU20 / HT6 as bt_conf signals an error for these devices.
- FB1 is configured by default in the PLC program (OB100 call), see *"FB9 description".*

Example:

[param MMC_1]
mmc_typ = 0x40
mmc_bustyp = BTSS
mmc_address = 1
mstt_address = **255**
name = MMC_Serv
start_mode = ONLINE

- Recommendation: The OPI/MPI addresses 0 (for PG) and 13 (for servicing purposes: replace NC) should be kept free.
- OFFLINE mode for MMC103 / PCU 50 : A server cannot be configured with boot property start_mode = Offline.
  If a main or secondary control panel is to be booted in offline mode, the following setting should be entered in the MMC.INI file.
  Enter the following setting in the [Global] section
  NcddeDefaultMachineName = LOCAL.

  After you do this, you should not select "Save" in the "Operator panel front interface parameters" menu, otherwise this entry will be overwritten again.

**HT6 removal/insertion**

The following are required for fault-free insertion and removal of the HT 6 **while the machine is running**:

- Release or override of the HT 6 EMERGENCY STOP.
- Connection of the HT 6 to the OPI/MPI via a PROFIBUS repeater.

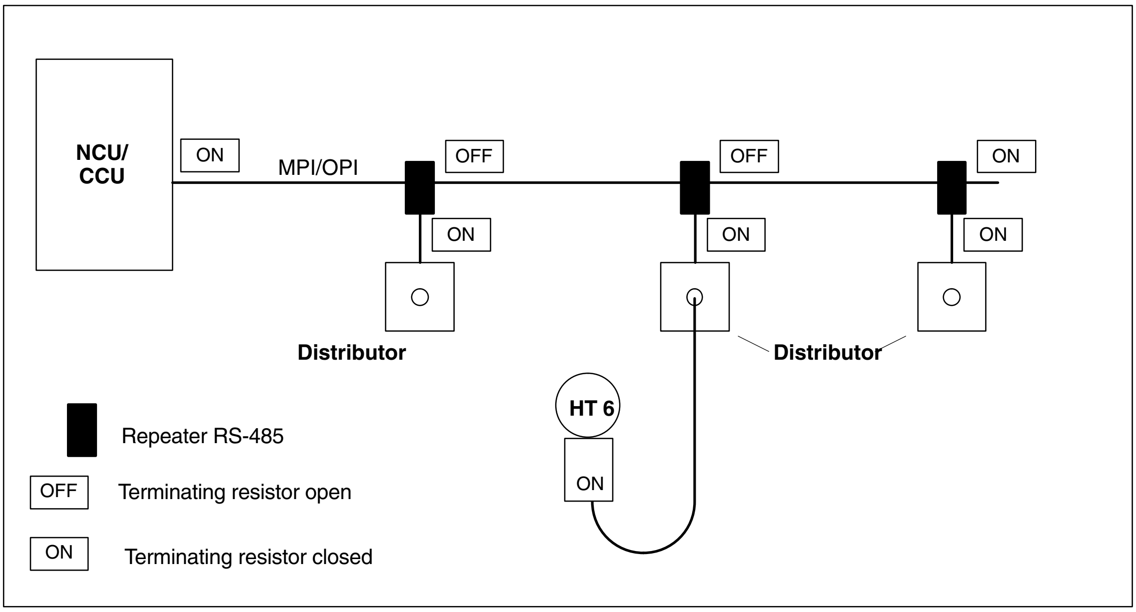



Fig. 6-16 Connecting the HT 6 using a PROFIBUS repeater

A PROFIBUS repeater must be connected upstream of the HT 6 distributor box for each branch. The individual bus segments (MPI/OPI cable and/or the local segments between repeater and HT 6) must be terminated with connector resistors at the ends of the bus.

**Repeater RS-485**

The repeater can be ordered under Order No. 6ES7972-0AA01-0XA0. For further information, please refer to the Catalog
/IK10/ Industrial Communication Networks SIMATIC-NET

---

**Note**

- The HT 6 already has an installed bus terminating resistor.
- The cable length from the repeater to the distributor box must not exceed 2 m.

---

You can find suggested circuits for the emergency stop in:

**References:** /BH/, Operator Components Manual.

## 6.5 Link axis

**Assumption**

NCU1 and NCU2 have one link axis each,
machine data e.g.:

```
; Machine data of NCU1:
$MN_NCU_LINKNO = 1                   ; Set NCU number to 1
                                     ; (master NCU)
$MN_MM_NCU_LINK_MASK = 1             ; Activate link function
$MN_MM_SERVO_FIFO_SIZE = 3           ; Size of data buffer 1)
                                     ; between interpolation
                                     ; and servo loop
$MN_MM_LINK_NUM_OF_MODULES = 2  ; Number of link modules

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX3"       ; Link axis

; Unique NCU axis names
$MN_AXCONF_MACHAX_NAME_TAB[0] = "NC1_A1"
$MN_AXCONF_MACHAX_NAME_TAB[1] = "NC1_A2"
$MN_AXCONF_MACHAX_NAME_TAB[2] = "NC1_A3"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0] = 1
$MC_AXCONF_MACHAX_USED[1] = 2
$MC_AXCONF_MACHAX_USED[2] = 3

...

; Machine data of NCU2:
$MN_NCU_LINKNO = 2              ; Set NCU number to 2 (slave NCU)
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_SERVO_FIFO_SIZE = 3  ; 1)
$MN_MM_LINK_NUM_OF_MODULES = 2

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC1_AX3" ; Link axis

; Unique NCU axis names
$MN_AXCONF_MACHAX_NAME_TAB[0] = "NC2_A1"
$MN_AXCONF_MACHAX_NAME_TAB[1] = "NC2_A2"
$MN_AXCONF_MACHAX_NAME_TAB[2] = "NC2_A3"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0] = 1
$MC_AXCONF_MACHAX_USED[1] = 2
$MC_AXCONF_MACHAX_USED[2] = 3

...
```

1) With SW 5 the machine data is: MD 10087: SERVO_FIFO_SIZE.

# 6.6 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.

## 6.6.1 Axis container rotation without a parts program wait

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

## 6.6.2 Axis container rotation with an implicit parts program wait

| Channel 1 | Channel 2 | Comment |
|---|---|---|
| AXCTWE(C1) | Parts program ... | Channel 1 enables the axis container for rotation |
| Parts program **with** movement of a container axis | Parts program ... | Channel 1 **waits implicitly** for axis container rotation |
| | AXCTSWE(C1) | Channel 2 enables the axis container for rotation, container rotates. Channel 1 continues. |

## 6.6.3 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 |

# 6.7 Evaluating axis container system variables

## 6.7.1 Conditional branch

| Channel 1 | Comment |
|---|---|
| AXCTWE(CT1) | Channel 1 enables the axis container for rotation. |
| MARKER1: Part program **without** movement of a container axis | |
| IF $AC_AXCTSWA[CT1] == 1 GOTOB MARKE1 | **Conditional branch** dependent on completion of axis container rotation. |
| Part program **with** movement of a container axis | |

## 6.7.2 Static synchronized action with $AN_AXCTSWA

| Channel 1 | Comment |
|---|---|
| IDS =1 EVERY $AN_AXCTSWA[CT1] == 1 DO M99 | Static synchronized actions:<br>Always output auxiliary function M99 at the beginning of an axis container rotation. |
| | **References:** /FPSY/, FB Synchronized Actions |

## 6.7.3 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 check position**

```
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
                                    ; Delay 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 1st channel
```

**Example 3.1**
**Use internal wait**

```
M3 S100         ; Program axis container spindle again
                ; An internal wait for the end of the axis container rotation
                ; is implemented
```

**Example 3.2**
**Use internal wait**

```
x=IC(0)         ; Program axis container axis x again
                ; An internal wait for the end of the axis container rotation
                ; is implemented
```

**Example 3.3**
**Use internal wait**

```
AXCTSWE(CTL) ; If an axis container is enabled again for rotation,
             ; an internal wait for the end of the preceding
             ; axis container rotation is implemented
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.

---

## 6.8 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).
- Linkages:
  - 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.
  - Rotations of drums A and B are autonomous.
  - Rotations of drums A and B are limited to 270°
    (range and torsion of supply lines).

**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:
Slides 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.

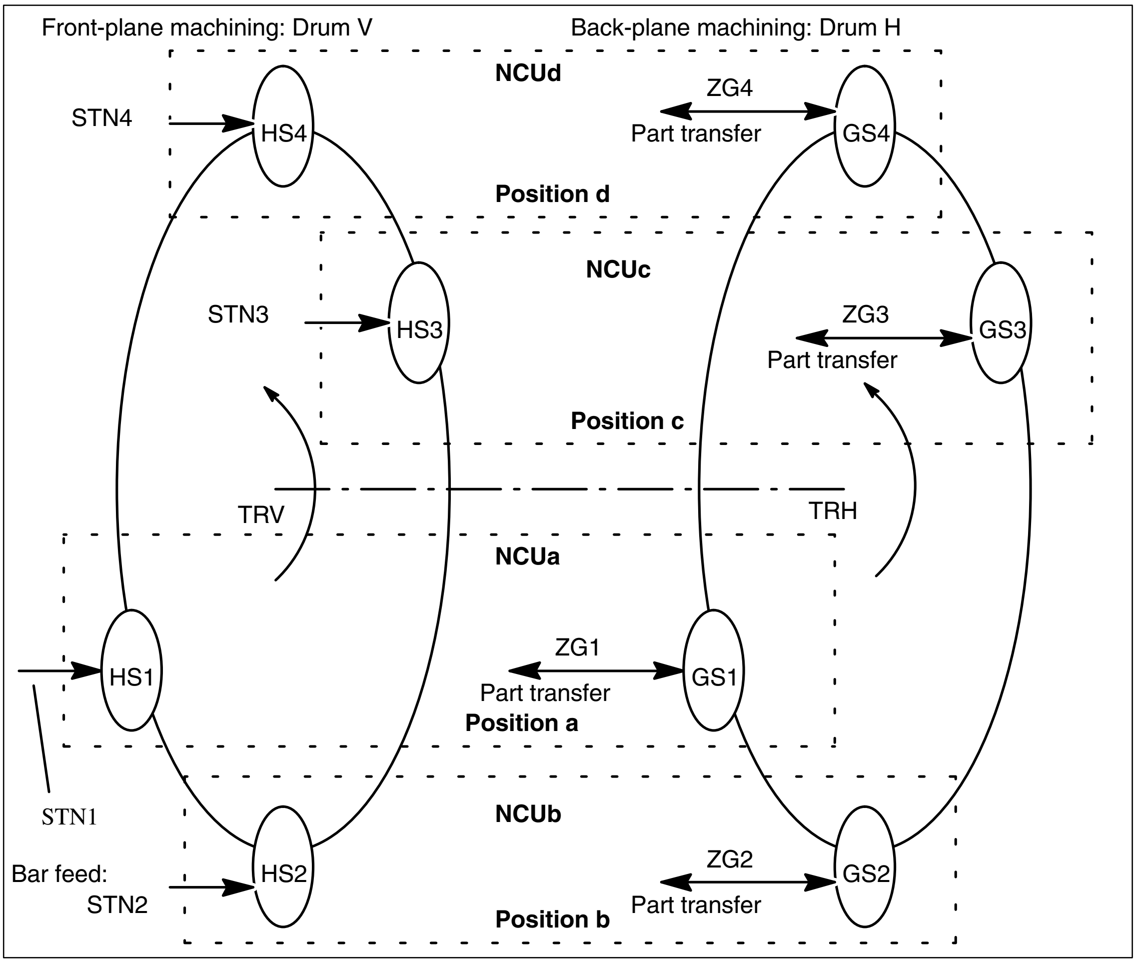



Fig. 6-17 Main spindles $HS_i$, countersp. $GS_i$, bar infeed axis $STN_i$ and transfer axes $ZG_i$ diagrammatic

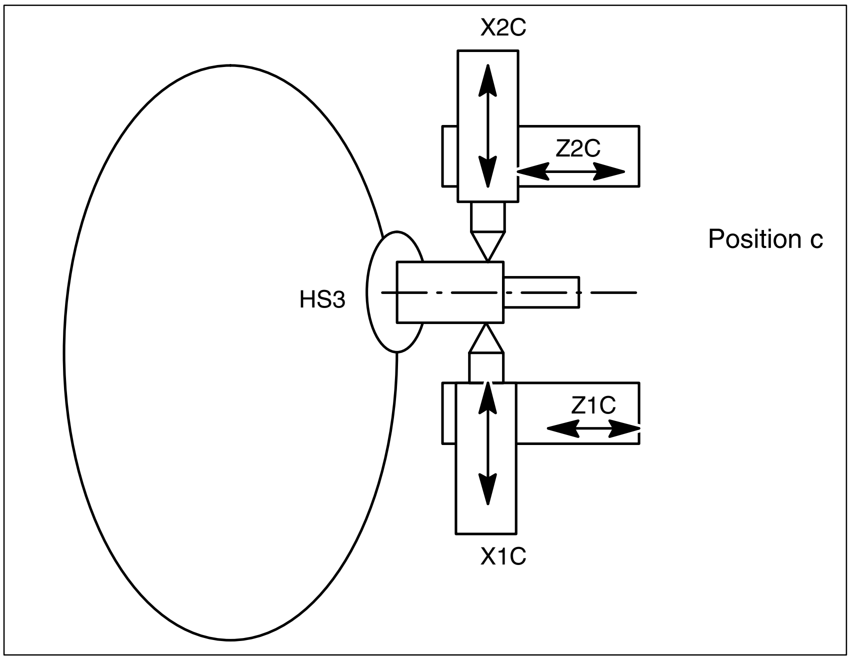



Fig. 6-18 Two slides per position can also operate together on one spindle.

**Note**

For clarifying the assignment of axes to slides and positions, the axes are named as follows:

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

Positions and their slides remain in a fixed position, whereas main spindles, counterspindles, bar feed axes STN and transfer axes ZG move to new positions by rotation 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 foregoing illustrations:

**Axes of the master NCU**

Table 6-5 Axes of master NCU: NCUa

| Common axes | Local axes | Remarks |
|---|---|---|
| | 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 Fig. “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 all assigned to a separate channel on an NCU.
  Thus slides can be moved autonomously.

**Configuration possibilities**

- Main or counterspindles are flexibly assigned to the slide.
- In each position the main spindle and counterspindle spindle speed can be determined independently.

  Exceptions:

  During part change from front-plane machining in drum V to rear-plane machining in drum H, the main spindle and counterspindle must be brought to the same spindle speed (synchronous spindle coupling).

  If slide 2 is also active in front-plane machining to “support” slide 1, in this case the main spindle speed is also valid for slide 2. By the same principle, if slide 1 is active in rear-plane machining, the counter spindle speed is also applicable for slide 1.

**Small changes in speed**

Owing 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 parts 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 exchange.

Table 6-6 NCUa, position: a, channel: 1, slide: 1

| Channel axis name | ..._MA-CHAX_USED | $MN_AXCONF_LOGIC_MA-CHAX_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 6-7 NCUa, position: a, channel: 2, slide: 2

| Channel axis name | ..._MA-CHAX_USED | $MN_AXCONF_LOGIC_MA-CHAX_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..” required 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:
  NCUb → 2
  NCUc → 3
  NCUd → 4
- The machine axis names are:
  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 6-6 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

MD 12701: $MN_AXCT_AXCONF_ASSIGN_TAB1[slot]
. . .
MD 12716: $MN_AXCT_AXCONF_ASSIGN_TAB16[slot]

with slots : 1 – 4 for the 4 positions of a multi-spindle turning machine:

---

**Note**

For the MD 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 6-8 Axis container and their position-dependent contents for drum A

| Container | Slot | Initial position (TRA o°) | 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° |

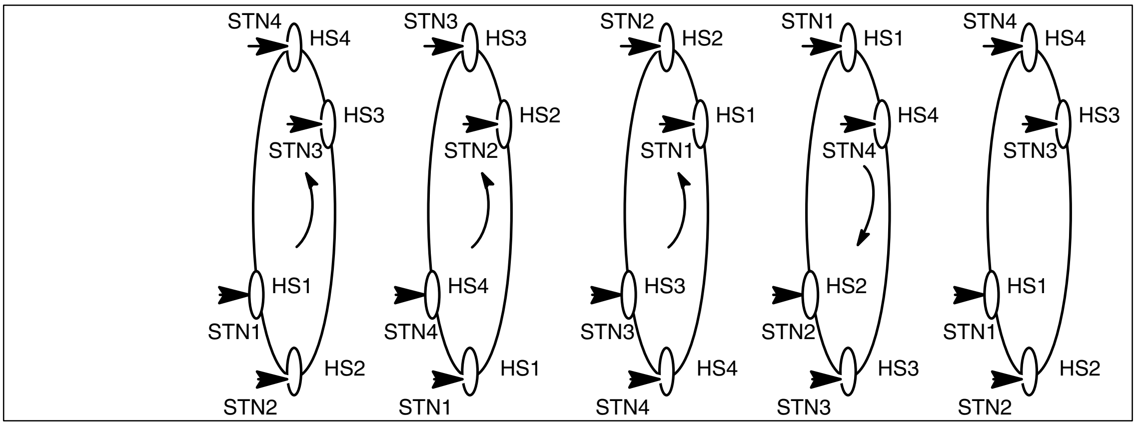



Fig. 6-19 Positions of drum A

Table 6-9 Axis container and their position-dependent contents for drum B

| Container | Slot | Initial position (TRB o°) | 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 |

# 6.9 Lead link axis

## 6.9.1 Configuration

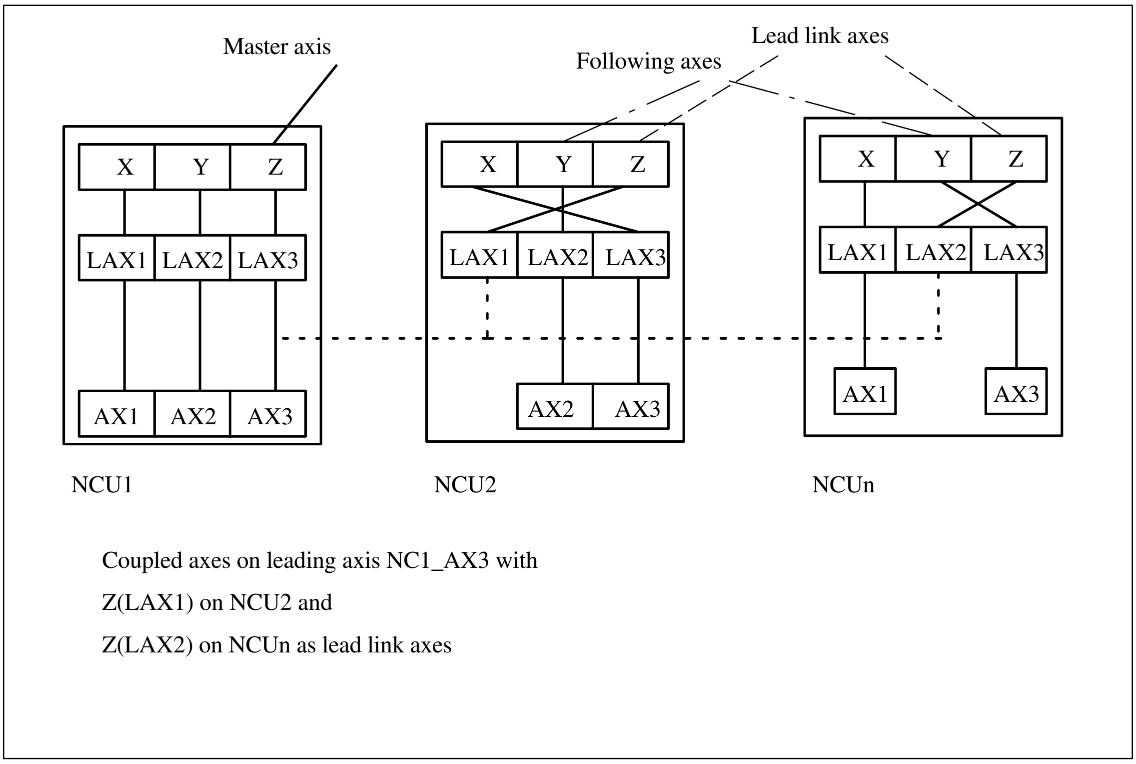



Fig. 6-20 NCU2 to NCUn use a lead link axis to enable coupling to the machine axis on NCU1 (NCU1–AX3).

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.

**Loading the machine data**

1. The machine data/setting data of a master value axis may only be loaded on the home NCU The machine data are distributed internally to the other NCUs where a lead link axis has been defined.
2. The lead link axis must be taken into account when configuring the NCU that is traversing the following axes (NCU2). The lead link axis occupies one location in the logical machine axis image (LAI) of the (NCU2). This reduces the maximum number of axes to be interpolated by this NCU by 1 for the lead link axis.
   In addition to the LAI axis location definition, the lead link axis must also be defined as channel axis ($MC_AXCONF_MACHAX_USED) in every channel where it will be used together with the following axis; this also reduces the maximum number of possible channel axes.

**Machine data for NCU1**

NCU traversing leading axis

```
$MN_NCU_LINKNO = 1                               ; 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                       ; Size of data buffer
                                                 ; between interpolation
                                                 ; and position control
                                                 ; increased to 4

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "AX3"
$MA_AXCONF_ASSIGN_MASTER_NCK[ AX3 ] = 1
$MN_AXCONF_MACHAX_NAME_TAB[0] = "XM1"
$MN_AXCONF_MACHAX_NAME_TAB[2] = "YM1"
CHANDATA(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**

NCU(s) traversing following axis

```
$MN_NCU_LINKNO = 2                               ; Set NCU number to 2
                                                 ;
$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 data buffer
                                                 ; between interpolation
                                                 ; and position control
                                                 ; (default)
$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

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=3   ; X
$MC_AXCONF_MACHAX_USED[1]=2   ; Y
$MC_AXCONF_MACHAX_USED[2]=1   ; Z               ; Assignment to LAI AX1
                                                 ; or NCU1/AX3
```

## 6.9.2 Programming

**Program on NCU 1**

NCU1 traverses leading axis Z. The variable is 1 for as long as NCU2 is prepared for movement of the leading axis (messages via link variable $A_DLB[0]); after completion of movement, the variable is 0.

```
N3000 R1 = 1                    ; Timer for movement loop
N3004 G1 Z0 F1000
N3005 $A_DLB[0] = 1             ; Start on NCU1
LOOP30:
N3005 R1=R1+1
N3006 G91 Z0.01                 ; The leading axis is now
                                ; traversed
N3008 Z0.02
N3010 Z0.03
N3012 IF R1 < 10 GOTOB LOOP30
N3098 $A_DLB[0] = 0             ; Terminate on NCU1
N3099 GOTOF TESTE
```

**NC program on NCU2**

The program establishes a connection between leading axis movements on NCU1 and following axis movements on NCU2 via a curve table. Once the table has been defined, NCU2 goes to wait position until NCU1 starts the leading axis. Then the coupling is activated and maintained until the leading axis movement is terminated.

```
N2800 CTABDEL(1)
N2801 G04 F.1
N2803 G0 Y0 Z0
;*****************************************************
; Create table 1
;*****************************************************
N2802 CTABDEF(Y, Z, 1, 0)
N2803 G1 X0 Y0
N2804 G1 X100 Y200
N2805 CTABEND
LOOP29:
N2806 IF ($A_DLB[0]== 0) GOTOB LOOP29 ; Wait for NCU1
N2810 LEADON(Y,Z,1)
LOOP292:                          ; Activate link !!!
N2830 IF ($A_DLB[0] > 0) GOTOB LOOP292 ; Maintain link until
                                  ;NCU1 is no longer traversing
                                  ;the leading axis
N2890 LEADOF(Y,Z)
```

# 6.10 NCU link with different interpolation cycles

## 6.10.1 Example of eccentric turning

**Task**

Create a non-circular shape with the following characteristics:

| | |
|---|---|
| Ellipticity: | 0.2 mm |
| Diameter of base circle: | 50 mm |
| Z path per revolution: | 0.1 mm |
| Spindle speed: | 3000 rpm |

A sinusoidal approximation via a cubic polynomial per 45 degrees of spindle revolution should be sufficient for the required precision.

---

**Note**

Polynomials up to the 5th degree can be used with software Version 6 and higher. See Programming Guide Advanced.
The part program (shown as an extract) is executed in a channel of the NCU with the faster interpolation cycle on which the X axis (local axis) and the C and Z axes are configured as link axes.
CAD systems are used to determine the polynomial coefficients by calculating them from points on the contour and the desired degree of polynomial.

---

The following parts program describes the commands required for the first spindle revolution. It must then be continued accordingly for the entire required length of the Z path:

```
G0 C0 X24.95 Z0                          ; Start position
FGROUP(C)                   ; Provides constant spindle speed
G1 G642 F1080000            ; Spindle speed 3000 RPM
POLY                        ; Specification of polynomials
C=DC(45.0000000) PO[X]=(25.0,.0750000,-0.0250000)
PO[Z]=(.2125000,0,0)
              ;1/8 circle, linear Z movement, 1/4 sine in X
C=DC(90.0000000) PO[X]=(25.0500000,0,-0.0250000)
PO[Z]=(.2250000,0,0)
C=DC(135.0000000) PO[X]=(25.0,-0.0750000,.0250000)
PO[Z]=(.2375000,0,0)
C=DC(180.0000000) PO[X]=(24.9500000,0,.0250000)
PO[Z]=(.2500000,0,0)
C=DC(225.0000000) PO[X]=(25.0,.0750000,-0.0250000)
PO[Z]=(.2625000,0,0)
C=DC(270.0000000) PO[X]=(25.0500000,0,-0.0250000)
PO[Z]=(.2750000,0,0)
C=DC(315.0000000) PO[X]=(25.0,-0.0750000,.0250000)
PO[Z]=(.2875000,0,0)
C=DC(0) PO[X]=(24.9500000,0,.0250000) PO[Z]=(.3000000,0,0)
...
```

**Machine data**

For information on configuring the machine data, please refer to 2.5.1, 2.6 and Section 4.2.

■

## Notes

# 7 Data Fields, Lists

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **General** | **Signals from NC to PLC** | | |
| 10 | 104.0 | MCP1 ready | |
| 10 | 104.1 | MCP2 ready | |
| 10 | 104.2 | HHU ready | |
| 10 | 107.6 | NCU link active | |
| 10 | 108.1 | MMC 2-CPU ready (MMC to OPI or MPI) | |
| 10 | 108.2 | MMC CPU1 Ready (MMC to MPI) | A2 |
| 10 | 108.3 | MMC CPU1 Ready (MMC to OPI, standard link) | A2 |

| DB number | Bit, byte | Name | | Reference |
|---|---|---|---|---|
| **General** | **Connection request indication interface** | | | |
| 19 | DBW100 | ONL_REQUEST | Online request from MMC | |
| 19 | DBW102 | ONL_CONFIRM | Acknowledgement to MMC | |
| 19 | DBW104 | PAR_CLIENT_IDENT | MMC bus address, bus type | |
| 19 | DBB106 | PAR_MMC_TYP | Main / secondary control panel / Alarm server | |
| 19 | DBB107 | PAR_MSTT_ADR | Address of MCP to be activated | |
| 19 | DBB108 | PAR_STATUS | Connection status | |
| 19 | DBB109 | PAR_Z_INFO | Additional information connection status / No. of the MMC-PLC interface | |
| 19 | DBW110 | M_TO_N_ALIVE | Ring counter, M:N switchover act. | |

| DB number | Bit, byte | Name | | Reference |
|---|---|---|---|---|
| **General** | **Online interface** | | | |
| 19 | DBW120 | MMC1_CLIENT_IDENT | MMC bus address, bus type | |
| 19 | DBB122 | MMC1_TYP | Main / secondary control panel / Alarm server | |

| DB number | Bit, byte | Name | | Reference |
|---|---|---|---|---|
| 19 | DBB123 | MMC1_MSTT_ADR | Address of MCP to be (de)activated | |
| 19 | DBB124 | MMC1_STATUS | Connection status | |
| 19 | DBB125 | MMC1_Z_INFO | Additional information connection | |
| 19 | DBX126.0 | MMC1_SHIFT_LOCK | MMC switchover disable | |
| 19 | DBX126.1 | MMC1_MSTT_SHIFT_LOCK | MCP switchover disable | |
| 19 | DBX126.2 | MMC1_ACTIVE_REQ | MMC requests active operating mode | |
| 19 | DBX126.3 | MMC1_ACTIVE_PERM | Enable from PLC to change the operating mode | |
| 19 | DBX126.4 | MMC1_ACTIVE_CHANGED | MMC has changed operating mode | |
| 19 | DBX126.5 | MMC1_CHANGE_DENIED | MMC active/passive switchover rejected | |
| 19 | DBW130 | MMC2_CLIENT_IDENT | MMC bus address, bus type | |
| 19 | DBB132 | MMC2_TYP | Main / secondary control panel / Alarm server | |
| 19 | DBB133 | MMC2_MSTT_ADR | Address of MCP to be (de)activated | |
| 19 | DBB134 | MMC2_STATUS | Connection status | |
| 19 | DBB135 | MMC2_Z_INFO | Additional information connection | |
| 19 | DBX136.0 | MMC2_SHIFT_LOCK | MMC switchover disable | |
| 19 | DBX136.1 | MMC2_MSTT_SHIFT_LOCK | MCP switchover disable | |
| 19 | DBX136.2 | MMC2_ACTIVE_REQ | MMC requests active operating mode | |
| 19 | DBX136.3 | MMC2_ACTIVE_PERM | Enable from PLC to change the operating mode | |
| 19 | DBX136.4 | MMC2_ACTIVE_CHANGED | MMC has changed operating mode | |
| 19 | DBX136.5 | MMC2_CHANGE_DENIED | MMC active/passive switchover rejected | |

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **General** | | **Signals from NC to PLC** | |
| 31 – 61 | 60.1 | NCU link axis active | |
| 31 – 61 | 61.1 | Axis container rotation active | |
| 31 – 61 | 61.2 | Axis ready | |

# 7.2 Machine/setting data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **General ($MN_ ...** | | | |
| 10002 | AXCONF_LOGIC_MACHAX_TAB[n] | Logical NCU machine axis image | |
| 10065 | POSCTRL_DESVAL_DELAY | Position setpoint delay | |
| 10087 | SERVO_FIFO_SIZE | Size of data buffer between interpolation and position controller task (up to SW 5, then MD 18720 see below) | |
| 10134 | MM_NUM_MMC_UNITS | Number of simultaneous MMC communication partners | |
| 11398 | AXIS_VAR_SERVER_SENSITIVE | Response of the AXIS-VAR server to errors | |
| 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 NCU link modules | |
| 12540 | LINK_BAUDRATE_SWITCH | Link bus baud rate | |
| 12550 | LINK_RETRY_CTR | Maximum number of message frame repeats in event of error | |
| 12701<br>...<br>12716 | AXCT_AXCONF_ASSIGN_TAB1[s]<br>...<br>AXCT_AXCONF_ASSIGN_TAB16[s] | List of axes in the axis container | |
| 12750 | AXCT_NAME_TAB[n] | List of axis container names | |
| 18700 | MM_SIZEOF_LINKVAR_DATA | Size of the NCU link variable memory | |
| 18720 | MM_SERVO_FIFO_SIZE | Size of data buffer between interpolation and position controller task (SW 6 and higher) | |
| 18780 | MM_NCU_LINK_MASK | Activation of NCU link communication | |

| | | | |
|---|---|---|---|
| **Channel ($MC_ ... )** | | | |
| 20000 | CHAN_NAME | Channel name | K1 |
| 20070 | AXCONF_MACHAX_USED | Machine axis number valid in channel | K2 |
| 28160 | MM_NUM_LINKVAR_ELEMENTS | Number of write elements for the NCU link variables | |

| | | | |
|---|---|---|---|
| **Axis ($MA_ ... )** | | | |
| 30550 | AXCONF_ASSIGN_MASTER_CHAN | Default assignment between an axis and a channel | K5 |
| 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 | |

| Setting data ($SN_ ... ) | | | |
|---|---|---|---|
| 41700 | AXCT_SWWIDTH[container number] | Axis container rotation setting | |
| 43300 | ASSIGN_FEED_PER_REV_SOURCE | Rotational feedrate for positioning axes/spindles | V1 |

## 7.3 Interrupts

A more detailed description of the alarms which may occur is given in

**References:** /DA/, Diagnostic Guide

or in the online help in systems with MMC 102/103.

■

# SINUMERIK 840D/840Di/810D
# Extended Functions
# Description of Functions (FB2)

# Operation via PG/PC (B4)



■

# Notes

# 1 Brief Description

**Applications**

The “Operation via PG/PC” functionality

- must be utilized if no operator panel front is installed.
- can be utilized as a handling support for OP030 panels.

**Hardware**

The following HW requirements must be fulfilled:

- PG/PC with at least 486DX33 processor and 8 MB main memory
- MS Windows must be running in ENHANCED mode (386 mode)
- PG/PC with MPI/OPI interface (provided with PG 720/720C/740/760).
  An MPI card (6ES7793-2AA00-0AA0) is available for PCs with free ISA slot.
- VGA monitor with a resolution of 640x480 or higher.

**Implementation with SW 3.1 and higher**

The option of creating a link between two operator panel fronts and one NCU has been implemented in SW 3.1 and higher. The machine control panel MCP is permanently allocated to the NCU.

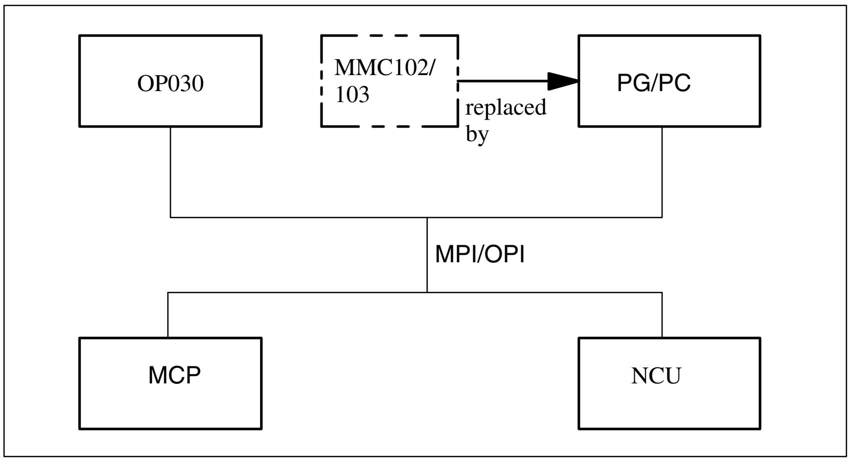



Fig. 1-1 Configuration with OP030 and PG/PC or MMC102/103

All operator panel fronts and the NCU are either connected to the OPI bus or all to the MPI bus. A **homogenous** network must be provided with respect to these components.

**Implementation with SW 3.2 and higher**

With software version 3.2 and higher, an additional option of linking one operator panel front and up to three NCUs is implemented. The machine control panel is permanently allocated to the NCU concerned.

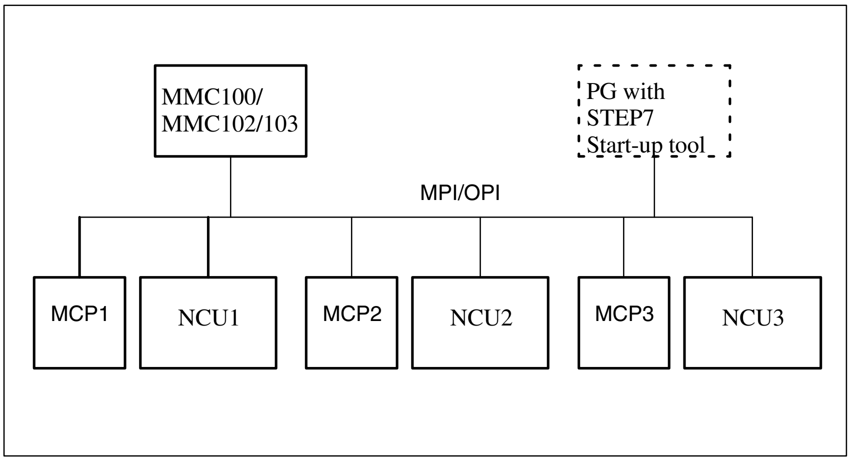



Fig. 1-2 Configuration in SW version 3.2 m:n corresponds to 1 : 3

**References:** /FB/ 2, B3, “Several Operator Panel Fronts and NCUs”

**Software installation**

see Chapter 2

**User interfaces**

The user interfaces are described in the Operator’s Guides of the relevant operator panel fronts.

**References:** /BA/, Operator’s Guide
/FBO/, BA, OP030 Operator’s Guide

**Restrictions**

If the “Operation via PG/PC” functionality is used in addition to an OP030 operator panel, the conditions relating to configuration and coordination described in /FB2/, B3, “Several operator panel fronts and NCUs” must be observed.

**References:** /FB/ 2, B3, “Several Operator Panel Fronts and NCUs”

■

# Detailed Description 2

## 2.1 Software installation

### 2.1.1 System requirements

**Hardware requirements**

The following hardware requirements must be fulfilled to allow operation via PG/PC:

- IBM® AT-compatible PG/PC with 486DX33 microprocessor
- At least 8MB of main memory
- Diskette drive (3 1/2 inch)
- Hard disk drive for managing data
- Monochrome or color monitor
- Keyboard
- PG/PC with MPI interface (available for PG 720/720C/740/760)
  Limited operation without MPI card is possible (e.g. interactive programming)
  **Note:** RS-232 MPI adapter is not supported.
- Mouse
- Connecting cable to link PG/PC and NCU module

---

**Note**

All operator panel fronts and the NCU are either
– connected to the OPI bus or
– connected to the MPI bus.
A **homogeneous** network must be provided with respect to these components.

---

**Software requirements**

Software configuration for operation via a PG/PC:

- MS-DOS operating system®, version 6.x or higher
- WINDOWS™ operator interface, version 3.1 or later
- MPI interface driver (contained in the supplied software).
- WINDOWS™ 32s, version 1.30.166.0 or later
  (you will find the current version in "windows\system\win32s.ini".)
  If WINDOWS™ 32s is not installed, it can be installed from 2 supplied diskettes (call setup.exe).

## 2.1.2 Installation

**Storage area of MPI card**

The storage area of the MPI card must be excluded from use by the memory manager (files: CONFIG.SYS, SYSTEM.INI).

**Example** of entry in SYSTEM.INI:
[386enh]
EmmExclude=....<card area>
(see HW description of card)

**Scope of delivery**

System software:

– Approx.10 diskettes with compressed MMC 102/103 software and installation tools

– 2 diskettes WINDOWS 32s subsystem (= Microsoft setup).

To install the software, please follow the procedure detailed below:

**Call-up**

1. **Start SETUP.EXE**
   Insert the first installation diskette and use the WINDOWS™ file manager to start the SETUP.EXE file.

   The installation program requests all further necessary inputs and disk changes in user dialog.

2. **Enter installation path**
   Select the directory plus the installation path (see screenshot) to which you wish to copy the Software.

   With **Continue**, you continue the installation, with **Exit Setup** you interrupt the installation procedure.
   This also applies to further operations.

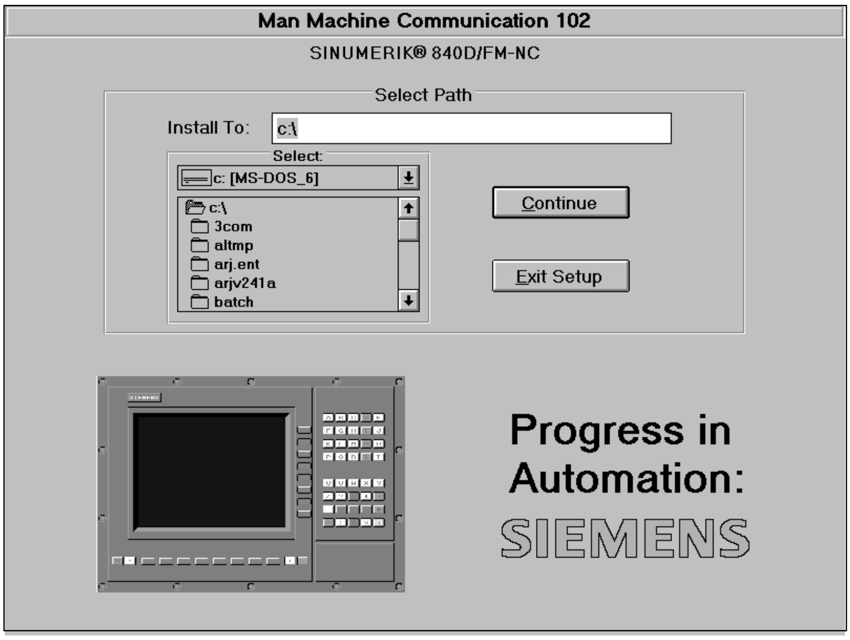



Fig. 2-1 Enter installation path

**3. Select operation with MPI or without MPI**


Fig. 2-2 Operation with/without MPI

**4.** Select **turning** or **milling**


Fig. 2-3 Select turning/milling

**Note**

If you want to change the selection mode later, select the directory “MMC 2” and copy “dpturn.exe” (turning) or “dpmill.exe” (milling) into the directory “dp.exe”.

**5.** **Select drive**,
only if several local disk drives are available
Select the drive for the tmp directory (see Fig.)

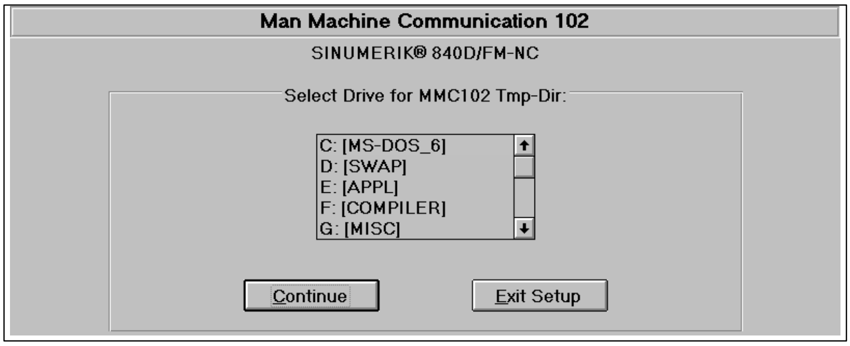



Fig. 2-4 Select drive

If this does not apply, select drive C:\.

**Note**

The contents of the directory “tmp” are deleted on the installation drive with each restart of MMC 102.

Following the selection, a status display with the inputs made is shown.

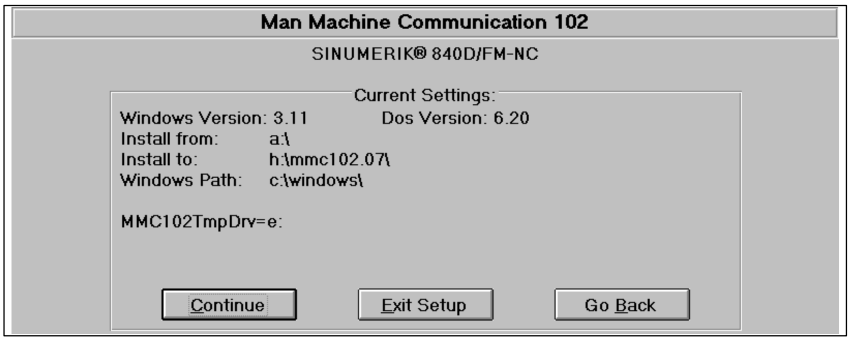



Fig. 2-5 Status display of the installation mode

**6.** Press **Continue** to request the installation diskettes.

---

**Note**

Please observe the requests made on the screen.

---

The program group “SINUMERIK 840D MMC V3.2” is generated.
With successful installation, the following message is displayed:
“MMC 102 Installation is complete”

If you want to change the installation path, press **Go back**.

**7.** **Make settings**

**7 a** **OPI** interface (1.5 Mbaud), Configuration: **1 MMC** to **1 NCU** (on delivery)

Additional settings are not required.

**7 b** **MPI** interface (187.5 kbaud), Configuration: **1 MMC** to **1 NCU** (on delivery)

1. Determination of the NCK/PLC bus address
   – if PLC < SW 3.2, then NC address = 13
     PLC address = 2
   – if PLC $\succeq$ SW 3.2 and module PLC 314, then NC address = 13
     PLC address = 2
   – if PLC $\succeq$ SW 3.2 and module PLC 315, then NC address = 3
     PLC address = 2
2. Entering the addresses in files
   – File “S7CFGPGX.DAT”
     In the file “S7CFGPGX.DAT” on the MPI driver directory (<installation path>\MMC2\DRV.ID) the following entries must be adapted to the existing hardware configuration by means of an ASCII editor:

# **Interrupt setting**
"hwint_vector": Setting the interrupt for the MPI card. This interrupt may not be used by another card (e.g. network adapter).
**Default setting**: 10.

# **Settings for baud rate**
"baud rate", "tslot" and "tgap": Settings for the baud rate. These 3 settings must always be activated/deactivated together by removing/inserting the leading "#" (comment).
When the baud rate is changed, the setting "ADDRESS1=\PLC, 10000d01" for 1.5 Mbaud or "ADDRESS1=\PLC, 10000201 for 187.5 kbaud must also be adapted in section [840D] in file <installation path>\MMC2\MMC.ini.
**Default setting:** 1.5 MBaud.

– File "netnames.ini"
The following lines in the file must be changed:

| | |
|---|---|
| # bus = opi | must be replaced by = **mpi** |
| # nck_address = 13 | must be replaced by = **3** (if PLC ≥ SW3.2)<br>= 13 (if PLC < SW3.2) |
| # plc_address = 13 | must be replaced by = **2** |

**Parallel STEP7/AS300 application**

Installation in parallel with the STEP7/AS300 SW can give rise to problems. It may be necessary to reconfigure the drives and restart the system.

## 2.1.3 Software supplementary conditions

- **Function keys**
  The function keys may not be actuated in any of the displays until the display has fully built up.
- **Monochrome screen**
  When a monochrome screen is used, the colors used by the MMC must be adapted accordingly. For this purpose, select the color scheme "Monochrome" or "Mono positive" in display "Start-up\MMC\Color setting".
- **Easy parameterization**
  The display "Start-up\MMC\OPI parameters" can now be called even if there is no link to the NC kernel. This means that the OPI parameters for baud rate and network address can be set easily.

### 2.1.4 Start the program

**Program call**

The MMC 102/103 software is started on a PG/PC either

- from the program manager through selection of the “SINUMERIK 840D MMC V2.3” program group followed by a **double click** on the **“MMC Startup”** symbol or


Fig. 2-6 SINUMERIK 840D MMC program group

- from the file manager by a **double click** on file **REG_CMD.EXE**.

**Communication**

If no communication link can be established to the NCK or 611D, then the message “No communication to NCK” is displayed. If the data exchange is interrupted, e.g. by an NCK reset, then the MMC 102/103 software tries to re-establish the communication link itself.

### 2.1.5 Close the program

**Deselecting the program**

The following steps must be taken to deselect the MMC 102/103 software:

1. Press function key **F10**
   A horizontal soft key bar is displayed.
2. Press function key **Shift** + **F9**
3. You can terminate the program by activating the **Exit** soft key.

## 2.2 Operation via PG/PC

### 2.2.1 General operation

**Operating philosophy**

The special function keys of the operator keyboard can be used with the full keyboard. Operator inputs can be made using the mouse or via the keyboard.

**Key assignments**

The following table shows the assignments between the function keys and the soft keys/special keys:

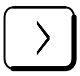
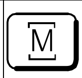
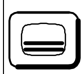
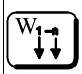
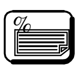
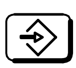
---

**Note**

The editor displays only the characters which can be input via the operator panel front keyboard.

---

Table 2-1 Key assignments between operator keyboard and full keyboard

| **Full key-board** | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| with SHIFT | **vertic** soft. **1** | **vertic** soft. **2** | **vertic** soft. **3** | **vertic** soft. **4** | **vertic** soft. **5** | **vertic** soft. **6** | **vertic** soft. **7** | **vertic** soft. **8** |  |  | | |
| without SHIFT | **horiz** soft. **1** | **horiz** soft. **2** | **horiz** soft. **3** | **horiz** soft. **4** | **horiz** soft. **5** | **horiz** soft. **6** | **horiz** soft. **7** | **horiz** soft. **8** |  |  |  |  |
| **Full key-board:** | 5 | Esc | Insert | Home | Page Up | Page Down | Enter | | | | | |
| **without SHIFT** |  |  |  |  |  |  |  | | | | | |

**Alarm or message line**

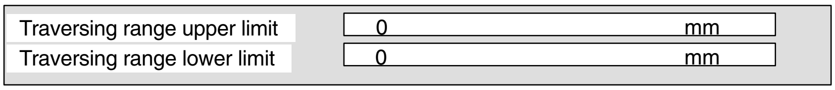

Alarm or message line for displaying information for the operator.

**i-R**

Selection fields **i** and **R** which appear in every display have the following meaning:

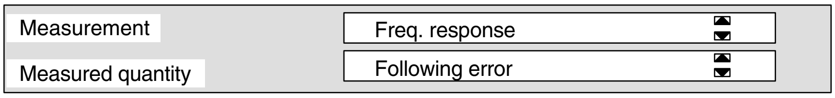
- The **i** field is selected with the **Help key** or by a **mouse click**. No help information is available in Software Version 1.
- The **R** field is selected with the key **F9** or by a **mouse click**. Selection of this field activates the Recall function, i.e. returns the user to the preceding level.

**Input fields**



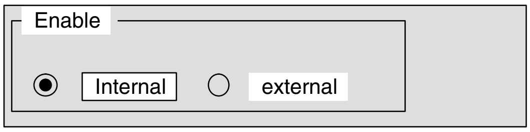

To allow the input of data, the input cursor is positioned in the appropriate input field by means of the **TAB** or **SHIFT + TAB** keys or by a **mouse click**. The editing mode is always preset to **Overwrite**. It is possible to switch back and forth between overwrite mode and insert mode by means of the **Insert** key.

**List fields**

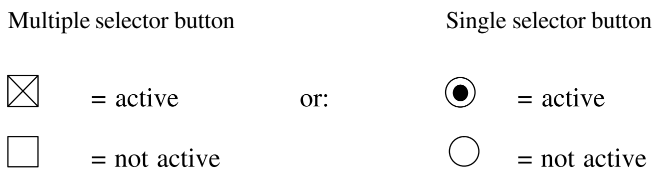



The functions offered are selected with the cursor keys **UP (↑)** and **DOWN (↓)** or by a **mouse click**. The displayed function is valid.
The list fields are selected by means of the **TAB** or **SHIFT + TAB** keys or a **mouse click**.

**Single/multiple selector button**




The required function is activated with cursor keys **LEFT (←)** and **RIGHT (→)** or by a **mouse click**.
The function fields are selected by means of the **TAB** or **SHIFT + TAB** keys or by a **mouse click**.




**Activation of fields**

To be able to alter values and functions, the window with the input field must be activated by means of keys **CTRL + TAB** or with the key **HOME** (yellow frame = focus).

### 2.2.2 Additional information

**Axis selection**

The “Select axis/Select next axis” inputs in axis-specific displays are always made via the uniformly positioned vertical soft keys **AXIS+** or **AXIS–**.

**Function selection/deselection**

All functions are activated by means of soft key **START** and deactivated by means of soft key **STOP**.

**Password**

When the soft key **Set password** is selected, a dialog box is displayed into which the password can be entered. Passwords are input as described in:

**References:** /BA/, Operator’s Guide
/FB/, A2, “Various Interface Signals”

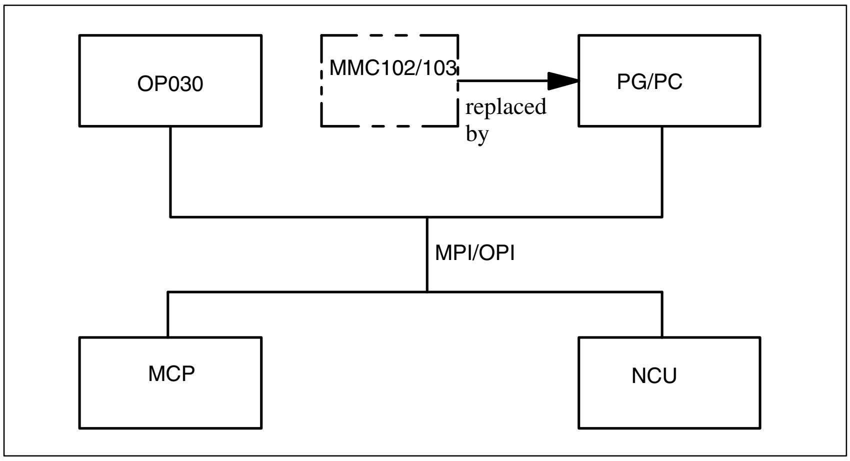
**Keyboard assignments**

Apart from the assignments of keys **F1** to **F12** and **SHIFT + F1** to **F10**, the conditions and key assignments are the same as those under WINDOWS™ 3.1.

The key combination **ALT + TAB** can be selected at any time to switch from “Operation via PG/PC” to other WINDOWS™ applications.

### 2.2.3 Operation of operator panel fronts

The system responds as follows, for example, when two panel fronts are operated in the configuration illustrated below:




1. For the NCU, there is no difference whether the input is from the MMC or OP030 operator panel fronts.
2. The operator panels are mutually independent in terms of data display, i.e. the display selected on one panel is not affected by the display on the other.
3. Spontaneous events such as alarms are displayed on both control units.
4. The protection level with the highest authorization in accordance with the lowest activated protection level number applies to both operator panel fronts.
5. The system does not provide for any further co-ordination between the operator panels.

For further information, please refer to
**References:** /FB/, B3, Several Operator Panel Fronts and NCUs
/BA/ Operator's Guide

## 2.3 Simulation of parts programs

A Windows 32s, version 1.30.166.0 or higher, must be installed in order to operate the parts program simulation.

For operating instructions, please refer to
**References:** /BA/ Operator's Guide.

■

# Notes

# 3 Supplementary Conditions

**Availability of the function**

The “Operation via PG/PC” function is available in the basic version with SW 3.1 and higher. With SW 3.1, the number of NCUs which may be connected is limited to one and the number of operator panel fronts to two. One of these must be an OP030.
With SW 3.2 and higher, an operator panel front with MMC 100 or MMC 102/103 can also be connected with up to three NCUs.

■

# 4 Data Descriptions (MD, SD)

No special machine data exist for this function.

■

## Notes

# 5 Signal Descriptions

No signals are required at the NCK–PLC interface for this function.

■

# 6 Example

None

■

# 7 Data Fields, Lists

No signals or machine data are required for this function.

## 7.1 Interrupts

A more detailed description of the alarms which may occur is given in

**References:** /DA/, Diagnostic Guide

or in the online help.

■

# Notes

# SINUMERIK 840D/840Di/810D
# Extended Functions
# Description of Functions (FB2)

# Remote Diagnostics (F3)

The Description of Functions **Remote Diagnostics** is included in the following documentation:

**/FBFE/** SINUMERIK 840D/840Di/810D
Description of Functions **Remote Diagnostics**

The previous document “Remote Diagnostics (F3)” in this manual has been replaced by the above document.

# Notes

# SINUMERIK 840D/840Di/810D
# Description of Functions
# Extended Functions (FB2)

# Manual and Handwheel Travel (H1)

■

# 1 Brief Description

**Setting up the machine**

Even on modern, numerically controlled machine tools, a facility must be provided that allows the operator to traverse the axes manually. 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, handwheel jogging can even be performed in the transformed workpiece coordinate system.

**Retraction of tool**

After a program interruption caused, for example, by NC-STOP, RESET or power failure, the machine operator must retract the tool manually 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.

**Contents**

The following Description of Functions illustrates the options and characteristics associated with the JOG traverse mode.

- Continuous jogging in jog or continuous mode (in JOG)
- Incremental jogging (INC) in jog or momentary-trigger mode (in JOG)
- Traversing the axes using electronic handwheels (accessories) (in JOG)
- Handwheel override in AUTOMATIC (path setting and velocity override)

**DRF**

The differential resolver function generates an additional incremental zero offset in AUTOMATIC mode via the electronic handwheel. This function can be used, for example, to compensate for tool wear within a programmed block.

■

# Notes

# Detailed Description 2

## 2.1 General

### 2.1.1 General characteristics of manual traverse in JOG

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

**JOG mode**

JOG mode must be active if the axes are to be traversed manually (referred to as “Manual traverse” below). The PLC receives the interface signal “Active mode: JOG” (DB11, DBX4.2) when the operating mode is activated.

**References:** /FB/, K1, “Mode Group, Channels, Program Operation Mode”

**Machine functions**

There are several JOG variants (so-called “machine functions”) within the JOG mode:

– Continuous jogging (JOG CONT)

– Incremental jogging (JOG INC)

– Jogging with the handwheel

**Handwheel operation**

The handwheel operation is also active with the following functions:

– Operating mode JOG-REPOS for moving the geometry and machine axes

– Operating mode AUTOMATIC, for moving out a DRF displacement

– With path override

– When moving the backlash point of an oscillation.

The 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).

**Simultaneous traversal**

All axes can be traversed simultaneously in JOG.
There is no interpolation between several axes traversed simultaneously.

**Velocity**

The velocity for JOG traversal is determined by the following value settings depending on the feed mode:

- When linear feedrate (G94) is active
  (SD: JOG_REV_IS_ACTIVE (revolutional feedrate in JOG) = 0):
  - with general SD: JOG_SET_VELO (JOG velocity with G94) or for rotary axes
    with the general SD: JOG_ROT_AX_SET_VELO (JOG velocity for rotary axes)
  - or with axis-specific MD: JOG_VELO (jog velocity), only if SD: JOG_SET_VELO = 0.
- When revolutional feedrate (G95) is active
  (SD: JOG_REV_IS_ACTIVE (revolutional feedrate in JOG) = 1):
  - with general SD: JOG_REV_SET_VELO (JOG speed with G95)
  - or with axis-specific MD: JOG_REV_VELO (revolutional feedrate in JOG), only if SD: JOG_REV_SET_VELO = 0.

The default setting for feedrate velocity is mm/min and for revolutional feedrate rpm.

**Rapid traverse override**

If the rapid traverse override key is pressed at the same time as the traversing keys, then the movement is executed at the rapid traverse velocity set in axis-specific MD: JOG_VELO_RAPID (axis velocity in JOG mode with rapid traverse override) (or in the case of revolutional feedrate, set in MD: JOG_REV_VELO_RAPID).

**Feedrate override**

The velocity traversed in JOG can be controlled additionally via the axial feed override switch provided that IS “Axial feedrate override active” (DB31, ... DBX1.7) is set.
Percentages are assigned to the individual positions of the feed override switch via machine data.
The axis does not traverse with switch position 0% if “0” is entered in the associated machine data. IS “Axial feedrate override active” has no meaning at switch position 0%.
Instead of the position on the feedrate override switch (Gray code), the value in percent (0% to 200%) can be set by the PLC. Again, the selection is made via the machine data.

**References:** /FB/, V1, “Feedrates”

**Acceleration**

Acceleration in manual traverse mode also takes place according to a programmed acceleration characteristic. The acceleration characteristic applicable in JOG mode for a single axis is defined in MD: AX_JERK_DEFAULT (initial setting of axial jerk limitation).

**References:** /FB/, B2, “Acceleration”

**Display**

The JOG basic display appears on the screen when JOG mode is selected. This basic display contains the position, feedrate, spindle and tool values.

For information about the individual displayed values see:

**References:** /BA/, “Operator’s Guide”

**Coordinate systems**

The operator has the option of traversing axes in different coordinate systems in JOG mode. 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 JOG traversing mode, an axis can be traversed as either a machine axis (axis-specific) or as a geometry axis (channel-specific). The characteristics of the machine axes are dealt with in the following description. The special features of traversing geometry axes in JOG mode are described in more detail in Subsection 2.8.1.

**Manual traversal of spindle**

Spindles can also be traversed manually in JOG mode. Essentially the same conditions apply as for manual traverse of axes. Spindles can be traversed in JOG mode using the traverse keys continuously or incrementally, in continuous-trigger or momentary-trigger mode, or with 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 apply to the spindles. The special features of traversing spindles manually are described in more detail in Subsection 2.8.2.

### 2.1.2 Control of manual traverse functions via PLC interface

**MMC/NCK/PLC interface**

Most individual functions required for manual traversal in JOG are activated via the PLC user interface. The machine-control manufacturer can adapt the JOG mode functions to the machine tool through the PLC user program depending on the configuration of the NC system.

**Machine control panel**

The signals between the machine control panel and the individual PLC/NCK interface data blocks are transferred by the PLC user program on a machine-specific basis. The PLC user program defines the assignment of the direction keys on the machine control panel to the axis/spindle (machine axes, geometry axes) traversing keys.

The following machine control panel signals are of particular importance to manual traverse:

- JOG mode (selection)
- Machine function INC1 ...
- Direction keys
- Feedrate override and spindle speed override

For further information on signals sent from the machine control panel see:

**References:** /FB/, P3, "Basic PLC Program"

**Selection of machine function**

The machine functions available in JOG mode can be selected from the following locations:

- From the machine control panel (MCP) e.g. user DB interface
- From the PLC user program PLC/NCK interface

The PLC user program transfers the machine function pending at the machine control interface to the relevant PLC/NCK interface. In this case, the axis-specific NCK/PLC interface (DB31, ... signals see Subsection 5.3) must be used for a machine axis/spindle and the channel-specific NCK/PLC interface (DB21, ... signals see Subsection 5.2) for a geometry axis.

## 2.1.3 Control response at power ON, mode change, reset, block search, repositioning

Any reset yields an abort with braking ramp of a traversing movement triggered by handwheel operation.

**Selection of MCP**

The following example shows the sequence of operations for selecting the "continuous" machine function for a machine axis of the machine control panel.

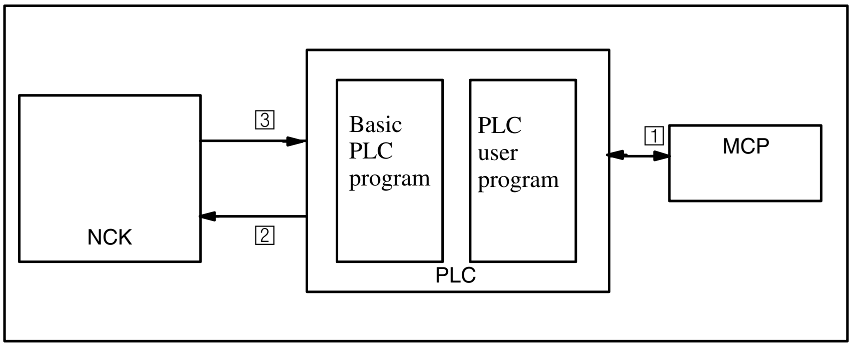



Fig. 2-1 Sequence of operation for selecting a machine function from the machine control panel

Sequence of operation:

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

2 IS "Machine function"
The PLC program (basic or user program) combines this IS and outputs the request to the NCK IS "Machine function continuous" (DB31, ... DBX5.6). The PLC user program checks first whether the request is permissible depending on the current machine status.

3 IS "Active machine function"
The control selects the machine function internally. The NCK notifies the PLC as soon as the machine function "Continuous JOG" (DB31, ... DBX65.6) becomes active.

For further information on signal transmission between the machine control panel and the PLC see:

**References:** /FB/, P3, "Basic PLC Program"

## 2.2 Continuous jogging

**Selection**

Continuous mode in JOG mode is selected via the PLC interface (IS “Machine function: Continuous” (DB21-28 DBX13.6, ff)). As soon as continuous mode is active, a signal is returned to the PLC with IS “Active machine function: Continuous” (DB21-28 DBX41.6, ff).

**Traversing keys +/–**

The “plus” and “minus” traversing keys are selected to move the relevant axis in the appropriate direction.
If both keys are pressed at the same time no movement takes place or a moving axis is stopped.

---

!

**Important**

When the control is switched on, axes can be traversed to the limits of the machine because they have not yet been referenced. Emergency limit switches might be triggered as a result.
The software limit switches and the working area limitation are not yet active.

---

**Travel command +/–**

As soon as a traverse request for an axis is active (e.g. after selection of a traverse key), the IS “Travel command+” or “Travel command–” (DB21, ... DBX40.7 or DBX40.6) is output depending on the direction of motion.

### 2.2.1 Distinction between jog mode and continuous mode

**Selection**

In JOG mode we distinguish between traversing in jog mode and continuous mode. The selection is made in the general SD: JOG_CONT_MODE_LEVELTRIGGRD (JOG continuous in jog mode) and is active for all axes.

**Default setting**

Jog mode is the default setting.

#### Continuous traversal in jog mode

**Function**

In jog mode (default setting) the axis traverses for as long as the traverse key is held down if no axis limitation is reached first. When the traverse key is released, the axis is decelerated to zero speed and the movement comes to an end.

## Continuous traversal in continuous mode

**Function**

When the traverse key is pressed and released (first rising edge) the axis starts to traverse at the set velocity in the desired direction. This movement is continued after the traverse key is released. The movement of the axis is either stopped by the operator or because of a response in the control (e.g. software limit switch reached).

---

**Warning**

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!

---

**Interrupting the traverse movement**

The operator can use the following methods to interrupt the traversing movement:

- Set feedrate override to 0%
- Axial feed disable (PLC interface signal)
- NC STOP or NC STOP axis/spindle

If the cause of the interruption is removed again, the axis continues to traverse.

**Aborting the traverse movement**

The traversing movement can be stopped and aborted by means of the following operator inputs or monitoring functions:

- Same traverse key pressed again (second rising edge)
- Traverse key for the opposite direction pressed
- RESET
- When continuous jogging is deselected
- On reaching the first valid limit

---

**Caution**

Software limit switches and working area limitations are only activated after reference point approach.

---

- When a fault occurs

---

**Note**

While an axis is moving, a change of mode from JOG to AUT or MDA is disabled internally.

---

### 2.2.2 Special features of continuous jogging

**Indexing axes**

When an axis that is declared as an indexing axis is traversed in continuous JOG mode, it always traverses to indexing positions. For example, the axis traverses on to the next indexing position in the direction of travel even if the key is released in jog mode.

**References:** /FB/, T1, “Indexing axes”

## 2.3 Incremental jogging (INC)

**Programming increments**

The traversing path to be traversed by the axis is defined by so-called increments (also termed “incremental dimensions”). Before the machine operator jogs the axis he must set the required increment.
The setting is made on the machine control panel, for example. The IS “Machine function: INC1 to INCvar” (DB31, ... DBB5 ff) associated with the desired increment must be gated and then set by the PLC user program.

**Settable increments**

The operator can set up to six different increment sizes. These are subdivided into:

- **Five fixed increments** whose increment sizes are defined jointly for all axes with the general MD: JOG_INCR_SIZE_TAB (increment size INC/handwheel). INC1, INC10, INC100, INC1000 and INC10000 are the default settings.
- **And a variable increment** (INCvar). The increment setting for the variable increment also applies jointly to all axes and is made in SD: JOG_VAR_INCR_SIZE (size of the variable increment for INC/handwheel).

**Increment weighting**

Use axial MD: JOG_INCR_WEIGHT (weighting of an increment of a machine axis for INC/handwheel) defines the path weighting of **one** JOG increment.

### 2.3.1 Distinction between jog mode and continuous mode

**Selection**

In incremental jogging, too, we distinguish between traversing in jog mode and continuous mode. The selection is made in the general MD: JOG_INC_MODE_LEVELTRIGGRD (INC and REF jog mode). Jog mode is the default setting.

## Incremental jogging in jog mode

**Function**

If the traverse key for the required direction (e.g. +) is pressed, the axis begins to traverse the increment set. If the traverse key is released before the increment has been traversed, the movement is interrupted and the axis stops. If the same key is pressed again, the axis moves the remaining distance until it is zero. As long as the remaining distance is greater than zero, the movement can again be interrupted by releasing the traverse key.
Pressing the key for the opposite direction has no effect until the increment has been completely traversed or the movement has been interrupted.

**Abort traversing movement**

If you do not want to traverse the whole increment, the traverse movement can be aborted with RESET or IS "Delete axial distance-to-go" (DB31, ... DBX2.2).

## Incremental jogging in continuous mode

**Function**

The axis traverses the entire set increment when the traverse key is pressed (on the first rising edge). If the same key is pressed again (second rising edge) before the axis has completed traversing the increment, the movement is aborted, i.e. not completed.

**Interrupt traversing movement**

As for continuous jogging.

**Abort traversing movement**

The traverse movement is stopped and aborted by the following operator action or monitoring functions:

– Same traverse key pressed again (second rising edge)
– Traverse key for the opposite direction pressed
– RESET
– Delete axial distance-to-go (PLC interface signal)
– On reaching the first valid limit

---

**Caution**

Software limit switches and working area limitations are only activated after reference point approach.

---

– On deselection or change of the current increment (e.g. change from INC100 to INC 10).
– On faults (e.g. on cancellation of the servo enable).

**Note**

While an axis is moving, a change of mode from JOG to AUT or MDA is disabled internally.

**Warning**

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!

## 2.3.2 Special features of incremental jogging

**Indexing axes**

Regardless of the currently set incremental value, the axis declared as an indexing axis (MD: INDEX_AX_ASSIGN_POS_TAB) traverses to the next highest indexing position after the traverse key “+” is pressed. Similarly, pressing the “–” traversing key causes the next lower indexing position to be approached.

**References:** /FB/, T1, “Indexing axes”

## 2.4 Handwheel traversal in JOG

**Selection**

JOG mode must be active. The operator must also set the increment INC1, INC10, ... which applies when jogging with the handwheel. As with incremental jogging the required machine function must be set at the PLC interface accordingly.

**Traversing**

When the electronic handwheel is turned the associated machine axis is traversed either in the positive or negative direction depending on the direction of rotation.

**Traversing path**

The traversing path produced by rotation of the handwheel is dependent on the following factors:

- Number of handwheel pulses received at the interface
- Active increment (machine function INC1, INC10, INC100, ... INCvar)
- Handwheel pulse weighting with general MD: HANDWH_IMP_PER_LATCH (handwheel pulses per detent position)
- Evaluation of an increment with INC/handwheel (axis-specific MD: JOG_INCR_WEIGHT).

**Travel command +/–**

IS “Travel command+” or “Travel command–” (DB31, ... DBX64.7 or DBX64.6) is output to the PLC while the axis is moving depending on the direction of motion.
If the axis is already being moved using the direction keys, the handwheel cannot be used. Alarm 20051 “Jogging with the handwheel not possible” is output.

**Handwheel connection**

Two handwheels can be connected simultaneously. In this way, up to two axes can be traversed by handwheel simultaneously.
Exception: If several axes are assigned to one handwheel, more than two axes can be traversed with handwheels.

**Handwheel assignment**

A separate axis-specific VDI interface signal is used to make the assignment between a handwheel and a geometry or machine axis.

The axis to be moved as a result of rotating handwheel 1 or 2 can be set:

- Via the PLC user interface with IS “Activate handwheel” (DB31, ... DBX4.0 – DBX4.2)
  (with a geometry axis: DB21, ... DBX12.0 – 12.2 ff)
- or by menu-guided operation (MMC)
  Operating the soft key **Handwheel** in the JOG mode basic menu displays the window “Handwheel”. Here, every handwheel can be assigned an axis and the handwheel can be enabled or disabled.

The assignment is transferred to the PLC interface through the logic of the PLC user program. In this way, several axes can be assigned to one handwheel simultaneously.

**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, handwheel jogging can even be performed in the transformed workpiece coordinate system.

**Handwheel selection from MMC**

A separate user interface between the MMC and PLC is provided to allow activation of the handwheel from the operator panel front. This interface that the basic PLC program supplies for handwheels 1 and 2 contains the following information:

- The axis numbers assigned to the handwheel IS "Axis number handwheel n" (DB10, DBB100 ff)
- Additional information on machine or geometry axis IS "Machine axis" (DB10, DBX100.7 ff)
- The channel number assigned to the handwheel if a geometry axis has been selected on handwheel selection IS "Channel number geometry axis handwheel n" (DB10, DBX97 ff).
- Information that the handwheel is enabled or disabled IS "Handwheel deselected" (DB10, DBX100.6 ff)

The IS "Activate handwheel" is either set to "0" (disable) or to "1" (enable) by the basic PLC program for the defined axis.

**Input frequency**

The handwheel connections can receive handwheel pulses with a maximum input frequency of 100kHz.

**Velocity**

The axis velocity settings SD:
JOG_SET_VELO (JOG velocity for G94) SD: JOG_ROT_AX_SET_VELO (JOG velocity for rotary axes) and MD: JOG_VELO (JOG axis velocity) used in JOG mode are also applied when axes are traversed manually.
Because of the limited feedrate, the axis is not able to follow the handwheel rotation synchronously, especially in the case of large pulse weighting, and therefore lags behind.

**Abortion of traversing movement**

A RESET command or IS "Axial delete distance-to-go" (DB31, ... DBX2.2) aborts the traversing motion. The setpoint/actual-value difference is deleted. STOP only interrupts the traversing movement. Any setpoint/actual-value difference is maintained. The distance-to-go is then traversed on START.

**Increment value limitation**

The operator can limit the size of the selected increment for geometry axes via the channel-specific machine data
(MD: HANDWH_GEOAX_MAX_INCR_SIZE).

The size of the selected increment for machine axes can be limited with axial machine data (MD: HANDWH_MAX_INCR_SIZE).

A traversing movement defined by the handwheel for a geometry axis is defined by

– Traverse path
– size of the variable increment (SD: JOG_VAR_INCR_SIZE)
– geometry axis allocation
  (MD: HANDWH_GEOAX_MAX_INCR_SIZE)

or for a machine axis by

– Traverse path
– size of the variable increment (SD: JOG_VAR_INCR_SIZE)
– machine axis allocation
  (MD: HANDWH_MAX_INCR_SIZE)

**Movement in the opposite direction**

Depending on the machine data $MN_HANDWH_REVERSE, the behavior with a change of the traversing direction is as follows:

- 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 movement direction, the unit is decelerated and the end point is approached by movement in the opposite direction. If this is not the case, the newly calculated end point is approached immediately.
- 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 interpolation are ignored. That means, another movement takes place only after zero speed (setpoint side) of the axis (new function). This feature is available with SW 3.2 and higher.

**Acceleration**

The acceleration rate for handwheel traversal is determined by the acceleration characteristic programmed in MD: AX_JERK_DEFAULT (initial setting of axial jerk limitation).
**References:** /FB/, B2, “Acceleration”

**Response 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 appropriate alarm is output. Depending on the machine data $MN_HANDWH_REVERSE, the behavior is as follows (as long as the axis on the setpoint side has not yet reached the end point):

- The distance resulting from the handwheel pulses forms a fictitious end point which is used for the subsequent calculations: If this fictitious end point is positioned for example 10mm behind the limitation, 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 be performed immediately after a limitation, the fictitious distance-to-go can be deleted via delete distance-to-go or deselection of the handwheel allocation.
- All handwheel pulses leading to an end point behind the limitation are ignored. Any movement of the handwheel in the opposite direction results immediately in a movement in the opposite direction, i.e. away from the limitation. This feature is available with SW 3.2 and higher.

**Limitations**

The limitations are also active when jogging with the handwheel. For further information see Subsection 2.8.3.

**Revolutional feedrate**

In JOG mode the behavior of the axis/spindle also depends on the setting of setting data JOG_REV_IS_ACTIVE (revolutional feedrate when JOG active).

- If this setting data is active, an axis/spindle is always moved with revolutional feedrate MD JOG_REV_VELO (revolutional feedrate with JOG) or MD JOG_REV_VELO_RAPID (revolutional feedrate with JOG with rapid traverse overlay) depending on the master spindle.
- If the setting data is not active, the behavior of the axis/spindle depends on the setting data ASSIGN_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 JOG_FEED_PER_REV_SOURCE. (In the operating mode JOG, revolutional feedrate for geometry axes on which a frame with rotation is effective).

### 2.4.1 Traversing request

New system responses to traversing request signals have been added; these are described below.

New PLC signals: Traversing request + and traversing request –:

DB21–30 DBB40 bit 5 traversing request + Geometry axis 1
DB21–30 DBB40 bit 4 traversing request – Geometry axis 1
DB21–30 DBB46 bit 5 traversing request + Geometry axis 2
DB21–30 DBB46 bit 4 traversing request – Geometry axis 2
DB21–30 DBB52 bit 5 traversing request + Geometry axis 3
DB21–30 DBB52 bit 4 traversing request – Geometry axis 3

DB21–30 DBB332 bit 5 traversing request + Orientation axis 1
DB21–30 DBB332 bit 4 traversing request – Orientation axis 1
DB21–30 DBB336 bit 5 traversing request + Orientation axis 2
DB21–30 DBB336 bit 4 traversing request – Orientation axis 2
DB21–30 DBB340 bit 5 traversing request + Orientation axis 3
DB21–30 DBB340 bit 4 traversing request – Orientation axis 3

DB31–61 DBB64 bit 5 traversing request + from axis/spindle
DB31–61 DBB64 bit 4 traversing request – from axis/spindle

**With path default**

In JOG mode with handwheel and with a path default (MD 11346: HANDWH_TRUE_DISTANCE == 1 or == 3) and a stop condition, which is **not an abort criterion** is valid
(see MD 32084: HANDWH_STOP_COND or
MD 20624: HANDWH_CHAN_STOP_COND),
the PLC signals traversing request and travel command are output in the normal way, see Figs. 2-2 and 2-3.

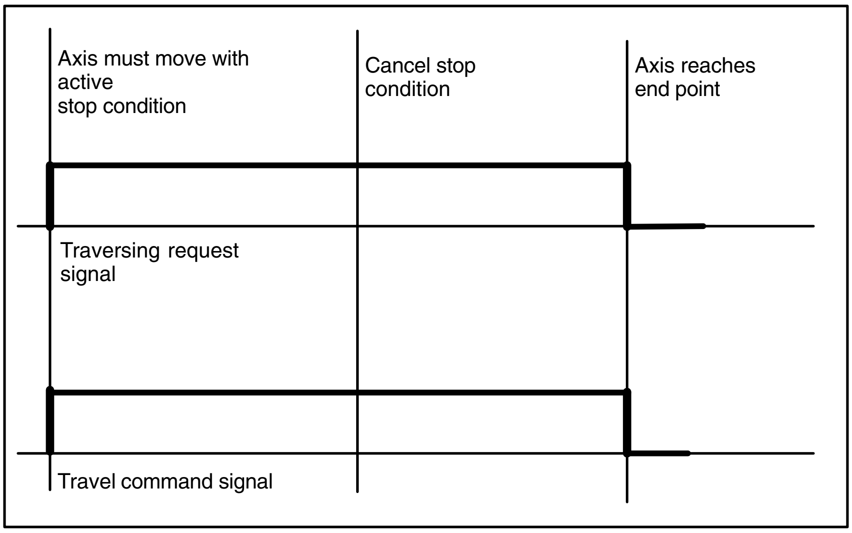



Fig. 2-2 Signal / time chart $MN_VDI_FUNCTION_MASK Bit 0 = 0

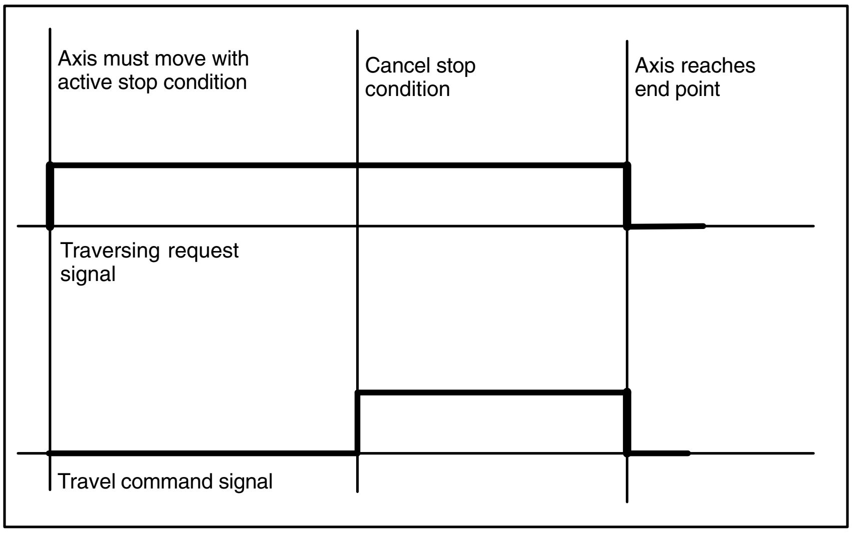



Fig. 2-3 Signal / time chart $MN_VDI_FUNCTION_MASK Bit 0 = 1

In JOG mode with handwheel and an active stop condition selected via machine data

MD 32084: HANDWH_STOP_COND or
MD 20624: HANDWH_CHAN_STOP_COND as the **abort criterion**, **no travel command** is output as in the previous software version (compatibility),

**but** the corresponding **traversing request is output.** When the stop condition is canceled, the corresponding PLC signal traversing request is reset, because the motion has been aborted. There is now no longer any stop condition active, but the axis cannot move because the stop condition has caused a motion abort.

Furthermore, it is also true that a path default
(MD 11346 $MN_HANDWH_TRUE_DISTANCE == 1 or == 3) is active or that the handwheel is being moved continuously i.e. is supplying pulses (see Fig. 2-4).

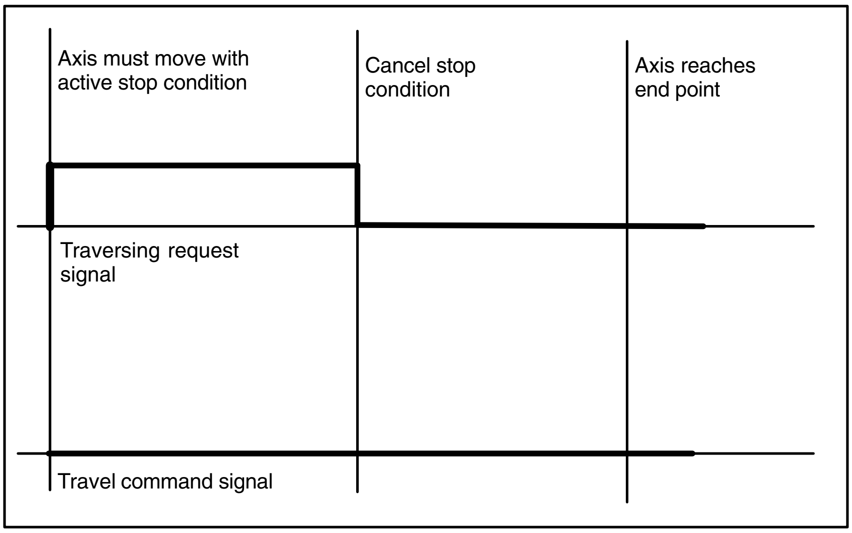



Fig. 2-4 Signal / time chart JOG with handwheel, stop condition is abort criterion

If a stop condition is activated in JOG mode with handwheel, the movement is aborted and the traversing request and travel command are reset.

**With velocity default**

If the handwheel is not moved again with a velocity default
(MD 11346: HANDWH_TRUE_DISTANCE == 0 or == 2),
the PLC signal traversing request is reset (see Fig. 2-5). The PLC signal traversing request is also reset when the handwheel is deselected.

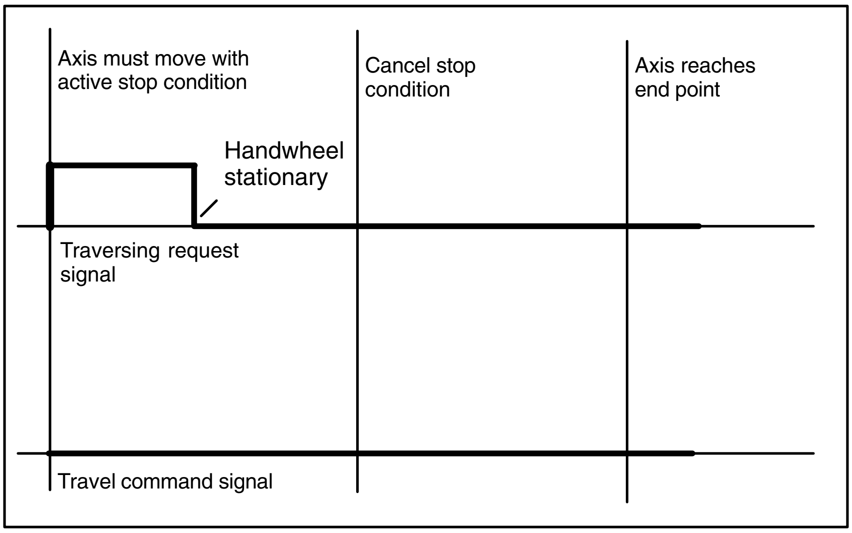



Fig. 2-5 Signal / time chart JOG with handwheel with velocity default, stop condition is abort criterion

**Supplementary conditions**

When NC Stop is active, no travel command and therefore no traversing request is output. This does not apply to DRF traversal:

If machine data MD 20624: HANDWH_CHAN_STOP_COND (bit 13 == 1) is set such that DRF traversal is permitted in the NC Stop state, the system response is the same as shown for JOG with handwheel in Figs. 2-4 and 2-5.

Analogous to the travel command calculation, the traversing request is determined from the sum of all partial movements, i.e. including the component from couplings and offset values.

**Examples**

Feed Stop is set as the abort criterion in machine data MD 32084: HANDWH_STOP_COND. The PLC signal Feed Stop is active. JOG with handwheel is selected (JOG mode, DRF traversal in AUTOMATIC mode).

The handwheel is rotated in the positive direction: The PLC signal traversing request + from axis/spindle is output, but no travel command + from axis/spindle.

The PLC signal Feed Stop is reset. No traversing request, no travel command.

### 2.4.2 Dual function of the handwheel

**Alarm 14320**

It is not permissible to use the handwheel for both DRF and velocity or path override (this also applies to the contour handwheel). Self-clearing alarm 14320 (Handwheel %1 used twice (%2) in channel %3 axis %4) will be activated if the handwheel is configured to perform two different functions in relation to the same axis.

This means that an overlaid motion cannot be performed until a DRF offset (initiated by the same handwheel) is no longer active in the basic coordinate system for any of the axes involved in the movement, in other words, the DRF motion must be finished.

If an overlaid motion is already in progress, it is not possible to start a DRF offset supplied by the same handwheel for any of the participating axes. In this case, the DRF motion cannot be initiated until the overlaid movement has reached its end point or has been aborted with a delete distance to go or a RESET command.

It is possible for a handwheel override and DRF offset to be active simultaneously provided they are supplied from **two** separate handwheels.

**Example: Path override**

Assumption:
Channel 1 and geometry axis X correspond to machine axis 3, geometry axis Y corresponds to machine axis 5 and
handwheel 2 is selected for the 1st geometry axis.

If block X10 Y10 FD=0 is executed in the main run, then 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 channel-specific signal DRF is active, alarm 14320 (Handwheel 2 used twice (8) in channel 1 axis X) is generated.

If machine axis 3 or machine axis 5 is traversed with DRF by means of handwheel 2, the motion X10 Y10 FD=0 cannot be executed and alarm 14320 (Handwheel 2 used twice (3) in channel 1 axis X) or 14320 (Handwheel 2 used twice (3) in channel 1 axis Y) is generated.

**Example velocity override on positioning axis**

Assumption:
Channel 1: Channel axis A corresponds to machine axis 4 and
handwheel 1 is assigned to machine axis 4.

If the block POS[A]=100 FDA[A]=0 is executed in the main run, then machine axis 4 cannot traverse with DRF, i.e. if the channel-specific signal DRF is active, alarm 14320 (Handwheel 1 used twice (6) in channel 1 axis A) is generated.

If machine axis 4 is traversing with DRF, then the motion programmed with block POS[A]=100 FDA[A]=0 cannot be executed while the DRF motion is still in progress. In this case, alarm 14320 (Handwheel 1 used twice (1) in channel 1 axis A) is generated.

**Example path override on PLC axis**

Assumption:
Channel 1: Handwheel 2 is assigned to machine axis 4.

If a block containing a motion with path override to be performed by the 4th machine axis and initiated by FC18 is executed in the main run, then machine axis 4 cannot traverse with DRF, i.e. if the channel-specific signal DRF is active, alarm 14320 (Handwheel 2 used twice (9) in channel 1 axis A) is generated.

If machine axis 4 is traversing with DRF, then the movement with path override initiated by FC18 cannot be executed while the DRF motion is still in progress. In this case, alarm 14320 (Handwheel 2 used twice (4) in channel 1 axis A) is generated.

## 2.5 Handwheel override in automatic mode

### 2.5.1 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 parts 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 the traverse instruction POSA. When the programmed target position is reached, the handwheel override is deactivated again.
Other axes can interpolate or traverse simultaneously in the same NC block.

The function for concurrent positioning axes can also be activated by the PLC user program.

**Variation**

Depending on the programmed feedrate, a distinction is made between the following:

- **Path definition** Axis feedrate = **0** (FDA = 0) and
- **Velocity override** Axis feedrate **> 0** (FD or FDA > 0)

Table 2-1 shows which axis types can be influenced by the function “handwheel override in Automatic mode”.

Table 2-1 Axes that can be influenced by the function “handwheel override in Automatic mode”

| Axis type | Velocity override | Path definition |
|---|---|---|
| Position axis | FDA[AXi] > 0 ;<br>axial | FDA[AXi] = 0 |
| Concurrent positioning axis | Parameter “handwheel override active” = 1 and axis feedrate > 0 from FC 15 | Parameter “handwheel override active” = 1 and axis feedrate = 0 from FC 15 |
| Path axis | FD > 0 ;<br>applies to path velocity | not possible |

**Path definition**

With an axis feedrate setting = 0 (e.g. when FDA[AXi] is programmed as 0), the traversing movement of the positioning axis towards the programmed target position is controlled entirely by the operator rotating the assigned handwheel.

The direction in which the handwheel is turned determines the direction in which the axis traverses. The programmed target position cannot be exceeded during handwheel override. The axis can also be moved in the direction opposite to that programmed, the movement in the opposite direction only being restricted by the axial position limits.

A block transition occurs

- when the axis has reached the programmed target position or
- the distance-to-go is deleted by the axial IS “Delete distance-to-go” (DB31, ... DBX2.2).

From this moment on, the path definition 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 feed and path feed.

- **Override of axis velocity** (FDA[AXi] > 0):
  The positioning axis is moved to the target position at the programmed axial feedrate. With the assigned handwheel it is possible to increase the axis velocity or to reduce it to a minimum of zero depending on the direction in which the handwheel is turned. The resulting axis feedrate is limited by the maximum velocity. The axis cannot be traversed in the direction away from the target position.
  As soon as the axis has reached the programmed target position, a block transition occurs. In this way, the velocity override is automatically deactivated again and any further handwheel pulses have no effect.
  This also applies to concurrent positioning axes, but the target position and the velocity are set by the PLC.
- **Override of path velocity** (FD > 0):
  The path axes programmed in the NC block move to the target position at the programmed path 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**. As soon as the programmed target position is reached, a block transition occurs.
  Depending on the direction in which the handwheel is turned, the path velocity is increased or reduced to a minimum of zero. It is not possible to reverse the direction of the movement with the handwheel override.

**Example of application**

The “Handwheel override in AUTOMATIC mode” function is frequently used on grinding machines. For example, the operator can position the reciprocated grinding wheel on the workpiece using the handwheel (path definition). After scratching, the traverse movement is terminated and the block transition is initiated (by activating “Delete distance-to-go”).

**Prerequisites**

In order to activate “Handwheel override in Automatic” mode the following conditions must have been met:

- A handwheel must be assigned to the axis in question.
- Pulse weighting is programmed for the assigned handwheel.

**Handwheel assignment**

The procedure for making the assignment between the connected handwheels and axes is analogous to the description in Section 2.4, i.e. via the PLC user interface with IS “Activate handwheel” (DB31, ... DBX4.0 bis DBX4.2) or on the basis of menu-assisted settings.
If a handwheel override is programmed for an axis to which no handwheel is assigned, the system will respond as follows:

- With velocity override:
  The axes are traversed at the programmed velocity. A self-acknowledging alarm is output (without response).
- With path definition:
  No traverse 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 traversing path of the axis that is generated by advancing the handwheel by one detent position is dependent on several factors (see Section 2.4).

- The selected increment size (general MD: JOG_INCR_SIZE_TAB[5] or axial SD: JOG_VAR_INCR_SIZE)
- The weighting of an increment (axial MD: JOG_INCR_WEIGHT)
- Number of handwheel pulses per detent position (general MD: HANDWH_IMP_PER_LATCH)

For example, the axis traverses by 0.001mm per handwheel detent position if machine function INC1 and the standard setting of the above machine data are selected.

With the velocity override, the velocity results from the path covered using the handwheel within a period of time.

Example

Assumptions:
The operator turns the handwheel at 100 pulses/second.
The selected machine function is INC100.
Machine data specified above for handwheel evaluation with default setting
⇒ Handwheel traversing 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 1:

- For positioning axes:
  IS “Handwheel override active” (DB31, ... DBX62.1)
- For concurrent positioning axes:
  IS “Handwheel override active” (DB31, ... DBX62.1)
- For path axes:
  IS “Handwheel override active” (DB21, ... DBX33.3)

In the case of a path override, the appropriate IS “Travel commands +/–” (DB31, ... DBX64.6 and 64.7) are output to the PLC depending on the direction of travel.

**Limitations**

The axial limitations (SW limit switches, HW limit switches, working area limitations) are effective in conjunction with handwheel override. With the path definition, the axis can be traversed with the handwheel in the programmed direction of travel only as far as the programmed target position.

The resulting velocity is limited by the axial MD: 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:

- With path definition:
  The handwheel pulses arriving in the meantime are summated and stored. On NC start or feedrate override > 0%, the stored handwheel pulses are activated (i.e. traversed).
  However, if the handwheel is deactivated first (IS “Activate handwheel n” DB21, ... DBB12/16/20), the stored handwheel pulses are deleted.
- With velocity specification:
  The handwheel pulses arriving in the meantime are not summated and are not active.

## 2.5.2 Programming and activation of handwheel override

**General**

When the handwheel override is programmed with the NC language elements **FD** (for path axes) and **FDA** (for positioning axes), the following points must be observed:

- FDA and FD are **nonmodal**.
  Exception with respect to positioning axes: If the traverse instruction POSA is programmed, the handwheel override can be active beyond block boundaries because the block transition is not affected by the positioning axis.
- 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 is deactivated 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] = [feed 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; the axis velocity of U is 100 mm/min |
| POSA[V]=20 | Target position of positioning axis V (beyond block limit) |
| FDA[V]=150 | Activate velocity override for positioning axis V; the axis velocity of V is 150 mm/min |

Example 2

Activate path definition 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 definition for positioning axis U; |
| POS [V]=200 | Target position of positioning axis V |
| FDA[V]=150 | Activate velocity override for positioning axis V; the axis velocity of V is 150 mm/min |

**Path axis** Syntax for handwheel override: **FD = [feed value]**

To program “Handwheel override in Automatic mode” for path axes, the following conditions must be fulfilled:

- Active movement commands from group 1: G01, G02, G03, CIP
- Exact stop active (G60)
- Linear feed in mm/min or inch/min active (G94)

These conditions 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; the path velocity is 1500 mm/min |

**Concurrent positioning axis**

The handwheel override for concurrent positioning axes is activated from the PLC via FC15. The appropriate parameter “Handwheel override ON” is set for this purpose.

If the parameter velocity (F_value) is assigned the value 0, the activated handwheel override acts as a path definition (i.e. the feedrate is not derived from the axial MD: POS_AX_VELO (initial setting for positioning axis velocity).

**References:** /FB/, P2, “Positioning Axes”
/FB/, P3, “Basic PLC program”

## 2.5.3 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 as diameter values and traversed as such while handwheel override is active.

**Dry run feedrate**

When dry run is active (IS “Activate dry run feedrate” (DB21, ... DBX0.6 = 1)), the dry run feedrate in (SD: DRY_RUN_FEED) always applies.
In this way, the axis is traversed at dry run feedrate without the handwheel influencing the programmed target position despite the active handwheel override with path definition (FDA[AXi]=0) (i.e. the path definition is not active).

**DRF active**

When “Handwheel override in automatic mode” is activated it is important to check whether the function DRF is active (IS “Activate DRF” (DB21, ... DBX0.3 = 1)). In this case, the handwheel pulses would also cause a DRF offset of the axis. The operator must therefore deactivate DRF first (see Section 2.9).

**Feedrate override**

The feedrate override does not affect the velocity of the movements produced by the handwheel (exception: 0%). It acts only on the programmed feedrate. With path definition and rapid jogging with handwheel, the axis might not follow the rotation of the handwheel synchronously (especially with a large handwheel pulse weighting), so that the axis lags.

## 2.6 Third handwheel via actual-value input (840D, 810D)

**Function**

**To date:**
It is possible to connect two handwheels to the peripheral interface (X121, 37-pin) on the NCU module using the cable distributor, etc.
**840D with SW 4.1 and higher, 810D with SW 2.1 and higher:**
It is now possible to connect a third handwheel via a 611D actual-value input that can be selected in a machine data.
A third handwheel could be used, for example, as a contour handwheel.

**Comparison of the three handwheels**

All three handwheels perform equally well in terms of response and functionality.
The third handwheel differs from the others only in terms of its connection method.

**Connecting the handwheel to the actual value input**

The signals of the handwheel (track A, *A, B, *B, 5V and 0V) must be wired to the actual value input as follows:

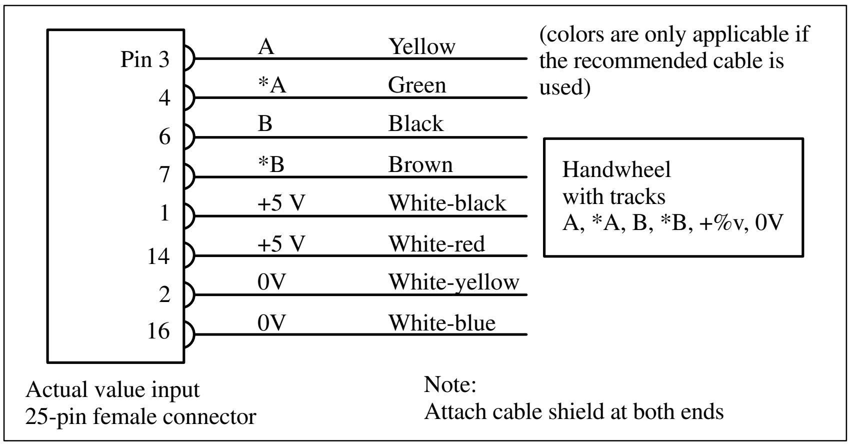



Fig. 2-6 Connecting the handwheel to the actual value input

**Recommendation:**

- Use the “Actual value cable for encoders with voltage signals” (6FX2002-2CG00- ...)
- Cut the cable at the circular connector, remove the outer shield and place on potential to ground
- Apply handwheel signals as shown in Fig. 2-6

**Activation, machine data and interface signals**

The following machine data and interface signals are required for activating the third handwheel:

- Machine data
  MD 11340: $MN_ENC_HANDWHEEL_SEGMENT_NR
  MD 11342: $MN_ENC_HANDWHEEL_MODULE_NR
  MD 11344: $MN_ENC_HANDWHEEL_INPUT_NR
- Interface signals
  IS “Channel number for handwheel 3” (DB10, DBX99.0, 99.1, 99.2)
  IS “Axis number for handwheel 3” (DB10, DBX102.0 to 102.4)
  IS “Define handwheel 3 as contour handwheel” (DB10, DBX102.5)
  IS “Handwheel 3 selected” (DB10, DBX102.6)
  IS “Machine axis” (DB10, DBX102.7)
  IS “Activate handwheel 3” (DB21, ... , DBX12.2, 16.2, 20.2)
  IS “Handwheel 3 active” (DB21, ... , DBX40.2, 46.2, 52.2)
  IS “Activate handwheel 3” (DB31, ... , DBX4.2)
  IS “Handwheel 3 active” (DB31, ... , DBX64.2)

**Supplementary conditions**

- The alarm “Handwheel %1 configuration faulty or inactive” is issued at power ON if any of the parameters for connecting the measuring circuit are incorrectly set or if hardware is missing.
- In contrast with actual value encoders, there is no encoder monitoring with connected handwheels. If hardware is defective or if cable break occurs, there are no handwheel impulses.
- There is no interlocking for dual assignment of an actual value input. Therefore, in principle, it is possible to assign an actual value encoder used for position/speed detection as “third handwheel” at the same time. In this case, the “handwheel pulses” are evaluated according to the number of encoder marks (course increments).
- The third handwheel can only be operated after successful power up of the SIMODRIVE 611D bus.

## 2.7 Contour handwheel / path definition by handwheel (840D, 810D)

**Function**

When the function is activated, the feedrate of path and synchronized axes can be controlled via a handwheel in AUTOMATIC and MDI mode.

**Operating characteristics of function**

The following operating characteristics in conjunction with the contour handwheel can be set in MD11346: HANDWH_TRUE_DISTANCE:

- Path definition:
  Limiting the velocity to the maximum permissible value causes the axes to overtravel. The path preset by the handwheel is traversed; **no pulses are lost**.
- Velocity specification:
  The handwheel specifies only the velocity at which the axes must traverse. As soon as the handwheel stops, the axes stop too. The axis motion is braked immediately, if no pulses are supplied by the handwheel in one IPO cycle, **preventing overtravel by the axes**. The handwheel pulses do not supply a path default.

**Feed**

The feedrate in mm/min is dependent upon:

- The number of pulses supplied by the selected handwheel within one period
- The pulse weighting of the handwheel via
  MD 11320: HANDWH_IMP_PER_LATCH.

  **In SW 5 and lower,** MD 11320 applies to both handwheel travel and contour handwheel travel. No separate pulse weighting and direction definition data was available for contour handwheel operation in this version.

  **In SW 6 and higher,** contour handwheel mode can be parameterized separately with MD 11322: CONTOURHANDWH_IMP_PER_LATCH.
- The activated increment (INC1, 10, 100, ...)
- The path weighting of an increment MD 31090: JOG_INCR_WEIGHT of first available geometry axis)

The feedrate is not dependent upon:

- The programmed feed mode (mm/min, mm/rev.)
- The programmed feedrate (resultant velocity can be higher)
- The rapid traverse rate with G0 blocks and
- The override (the position 0% is effective, i.e. zero speed)

**Traversing direction**

The travel direction is dependent upon the rotational direction:

- **In clockwise direction:** Causes axes to travel in the programmed direction If the block change criterion (IPO end) is reached, then the program advances to the next block (response identical to G60).
- **In counterclockwise direction:** Axis traverses in the programmed direction In this case, the axis can traverse only up to the start of the next block. Pulses are not collected if the handwheel continues to rotate.

**Activation of the function**

The function can be activated via the interface signals or via the NC program.

- Activation via the interface signal “Activate handwheel x as contour handwheel”
  The function is activated/deactivated by means of the following interface signal:
  IS “Activate handwheel x as contour handwheel” (DB21, ... DBX30.0, 30.1, 30.2)

- Activation via NC program
  The contour handwheel can be activated in the program for individual blocks with FD=0, that is, velocity F from the block before the contour handwheel is effective in the following block **without** any new programming.

---

**Note**

If the preceding NC blocks do not contain a feedrate, then an appropriate alarm is output.
FD and F in one NC block are mutually exclusive (e.g. generate an alarm).

---

**Simulation of the contour handwheel**

When the contour handwheel is activated, it can also be simulated. After activation via the interface signal, the feedrate is no longer preset by the contour handwheel; the programmed feedrate is used instead. The direction is also preset via interface signal.

IS “Simulation contour handwheel” (DB21, ... DBX30.3)
IS “Negative direction simulation contour handwheel” (DB21, ... DBX30.4)

| If | then |
|---|---|
| simulation is deselected, | the active movement is braked with a deceleration ramp |
| the direction is changed, | |
| **Note:** The override is effective as with NC program execution. | |

**Supplementary conditions**

- **Requirements**
  Permanent feed, dry run feedrate, thread cutting or tapping must not be activated.
- **Limit values**
  Acceleration and velocity of the axes are limited to the values defined in the machine data.
- **Interruption of traversing movement**
  The function remains selected after an NC STOP, but the handwheel pulses are no longer summated and are ineffective (on the condition, however, that in MD 32084: HANDWH_CHAN_STOP_COND bit 2 = 1).
- **DRF**
  In addition, a selected DRF function has a path overlay action.
- **Channel-specific deletion of distance-to-go**
  This causes the movement that was triggered by the contour handwheel to be aborted; the axes are braked and the program is restarted with the next NC block.
  The contour handwheel is then effective again.

# 2.8 Special features of JOG mode

## 2.8.1 Geometry axes in JOG mode

**Coordinate systems in JOG mode**

In JOG mode the operator can also jog the axes declared as geometry axes in the workpiece coordinate system. Any coordinate offsets or rotations that have been selected remain active.

---

**Note**

With SW 6.3 and higher of the SINUMERIK 840D with the “handling transformation package”,
the translation of geometry axes can be set separately in JOG mode in several valid reference systems.
**References** /FB/, F2, “Cartesian Manual Travel”

---

**Application**

Jogging movements for which transformations and frames have to be active. The geometry axes are traversed in the coordinate system that was last activated. The special features of jogging geometry axes are described below.

**Simultaneous traversal**

Only one geometry axis can be jogged continuously or incrementally using the traverse keys. Where an attempt is made to jog more than one geometry axis, alarm 20062 “Axis already active” is output. However, 3 geometry axes can be jogged simultaneously with handwheels 1, 2 and 3. Alarm 20060 is output if only one axis is not defined as a geometry axis.

**PLC interface**

A separate PLC interface (DB21, ... DBB12-23 and DBB40-56) exists for geometry axes that contains the same signals as the axis-specific PLC interface.

**Feedrate/rapid traverse override**

The channel-specific feedrate override switch and rapid traverse override switch are active for jogging geometry axes.

**Alarms**

Alarm 20062 “Axis already active” is triggered when a geometry axis is being jogged 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 traverse 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 move it in JOG mode.

## 2.8.2 Special features of spindle jogging

**Manual traversal of spindle**

Spindles can also be traversed manually in JOG mode. Essentially the same conditions apply as for manual traverse of axes. Spindles can be traversed in JOG mode using the traverse keys continuously or incrementally, in continuous-trigger or momentary-trigger mode, or with 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**

The spindle can be jogged in positioning mode (spindle is in position control) or in open-loop control mode.

**JOG velocity**

The speed used for jogging spindles can be defined as follows:

– With the general SD: JOG_SPIND_SET_VELO (JOG speed for spindle), which applies to all spindles,

– or with machine data JOG_VELO (JOG axis velocity) However, this MD only has an effect if SD: JOG_SET_VELO (JOG velocity for G94) = 0.

The maximum speed for the active gear stage also applies when spindles are traversed in JOG mode.

**References:** /FB/, S1, "Spindles"

**Speed override**

The spindle speed override switch can be used to modify the speed of spindles traversed in JOG mode.

**JOG acceleration**

Because a spindle often uses many gear stages in speed control and position control mode, the acceleration programmed for the current gear stage is always applied in spindle JOG mode.

**References:** /FB/, S1, "Spindles"

**PLC interface signals**

When spindles are traversed manually, the PLC interface signals between the NCK and PLC have the same effect as for machine axes. The IS "Position reached with exact stop fine/coarse" (DB31, ... DBX60.7 or DBX60.6) are set only if the spindle is in position control mode.
For purely spindle-specific interface signals, the following should be noted with respect to spindle traversal in JOG:

- The following PLC interface signals to the spindle have no effect:

  – IS "Invert M3/M4" (DB31, ... DBX17.6)

  – IS "Set direction of rotation CCW" or "Set direction of rotation CW" (DB31, ... DBX18.7 or DBX18.6)

  – IS "Oscillation speed" (DB31, ... DBX18.5)

  – IS "Spindle RESET" (DB31, ... DBX16.7)

- The following PLC interface signals from the spindle are not set:
  - IS “Actual direction of rotation CW” (DB31, ... DBX83.7)
  - IS “Spindle in setpoint range” (DB31, ... DBX83.5)

### 2.8.3 Monitoring functions

**Limitations**

The following limitations are active in JOG mode:

- 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 limitation 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 traverse keys and the handwheel have no effect in this direction.

---

!

**Important**

The software limit switches and working area limitations are only active if the axis is first referenced.

If a WO (DRF offset) is applied to an axis via handwheel, the software limit switches for the relevant axis are monitored by the main run in JOG mode. This means that the jerk limitation is not operative as the software limit switches are approached. After the axis has accelerated according to machine data MD 32300: MAX_AX_ACCEL, the velocity is reduced at the software limit switch.

---

For further information on working area limitations and hardware and software limit switches see:
**References:** /FB/, A3, “Axis Monitoring, Protection Zones”

**Axis retraction**

The axis can be retracted from a limitation position by moving it in the opposite direction.

**Machine Manufacturer**

The function for retracting an axis that has reached the limitation position depends on the machine-tool manufacturer. Please refer to the machine-tool manufacturer's documentation!

**Maximum velocity and acceleration**

The velocity and acceleration values applied in JOG mode are programmed for specific axes via machine data by the start-up engineer. The control limits the value for the values valid for the axes to the maximum velocity and acceleration specifications.

**References:** /FB/, G2, "Velocities, Setpoint/Actual-Value Systems, Closed-Loop Controls"
/FB/, B2, "Acceleration"

## 2.8.4 Miscellaneous

**Switching modes from JOG→AUT or from JOG→MDA**

It is possible to switch operating modes from JOG to AUT or MDA only if all axes in the channel have reached "Exact stop coarse".

**References:** /FB/, K1, "Mode Group, Channels, Program Operation Mode"

**Rotational feedrate active in JOG**

In JOG mode, it is also possible to traverse an axis at a rotational feedrate (analogous to G95) referred to the current speed of the master spindle. The function is activated with SD: JOG_REV_IS_ACTIVE (JOG in revolutional feedrate).
The feedrate value (in mm/rev) used can be defined as follows:

- With general SD: JOG_REV_SET_VELO (JOG speed for G95) if this is not equal to 0.
- If the value 0 is set in SD: JOG_REV_SET_VELO, then the rotational feedrate is determined by axial machine data JOG_REV_VELO (rotational feedrate for JOG) or, in the case of rapid traverse override, by JOG_REV_VELO_RAPID.

If a master spindle has not been defined and the axis is to be traversed in JOG at revolutional feedrate, alarm 20055 and for geometry axes alarm 20065 is output.

**Transverse axes**

If a geometry axis is defined as a transverse axis and radius programming is selected (MD: DIAMETER_AX_DEF (geometry axes with transverse axis function)), the following must be noted when traversing in JOG:

- Continuous jogging:
  There are no differences when a transverse axis is traversed in continuous mode.
- Incremental jogging:
  Only **half the distance** of the selected increment size is traversed. For example, with INC10 the axis only traverses 5 increments when the traverse key is pressed.
- Jogging with the handwheel:
  As for incremental jogging, only half the path is traversed per handwheel pulse.

**References:** /FB/, P1, “Transverse Axes”

## 2.9 DRF

**DRF function**

With the function DRF (Differential Resolver Function) an additional incremental zero offset can be activated with the electronic handwheel in automatic mode during machining.
The same conditions apply to the handwheel assignment, pulse weighting, etc., as for manual traverse with the handwheel in JOG mode (see Section 2.4). In addition, the velocity generated by the handwheel in DRF can be reduced from the JOG velocity with the axial MD: HAND_VELO_OVERLAY_FACTOR (ratio of JOG velocity to handwheel velocity).

**DRF offset**

The DRF offset is an axis zero offset that is generated by means of DRF (i.e. traversing movement with handwheel in automatic mode). The DRF offset is active in the basic coordinate system.

**References:** /BAD/ “HMI Advanced Operator’s Guide”
/BEM/, “HMI Embedded Operator’s Guide”

**Attention!**

Zero offset with DRF is always active; i.e. for all modes and after RESET. It can, however, be suppressed non-modally in the parts program.

**References:** /PG/, “Programming Guide: Fundamentals”

**Applications**

DRF is used for the following applications:

→ Compensating for tool wear within an NC block.
Where NC blocks have a long machining time it might be necessary to compensate for tool wear manually within the NC block (e.g. large surface milling machines).

→ Highly precise compensation during grinding

→ Very simple temperature compensation

→ Offsets that are not included in the actual-value display.

**DRF active**

DRF must be active to allow the DRF offset to be modified through traversal with the handwheel. The following conditions must be fulfilled:

- AUTOMATIC mode is selected (channel in RESET/interrupted/active)
- and IS “Activate DRF” (DB21, ... DBX0.3) = 1

DRF offset can be switched off for specific channels by the operator with the program control function. The MMC returns this state to the PLC using IS “DRF selected” (DB21, ... DBX24.3). The PLC program (basic PLC program or user program) transfers this interface signal as IS “Activate DRF” after the corresponding logic operation.

**Control of DRF offset**

The DRF offset can be modified, deleted or read in the following ways (see Fig. 2-7):

- By the operator by jogging with the handwheel
- By the NC parts program (in high-level language)
  - Reading the DRF offset (axis-specific)
  - Deleting the DRF offset for all axes in a channel (command “DRFOF”).

**References:** /PG/, “Programming Guide: Fundamentals”

- From PLC user program
  - Reading the DRF offset (axis-specific)

**References:** /FB/, P3,

“Basic PLC Program”

- From the MMC by the operator
  - Reading the DRF offset (axis-specific)

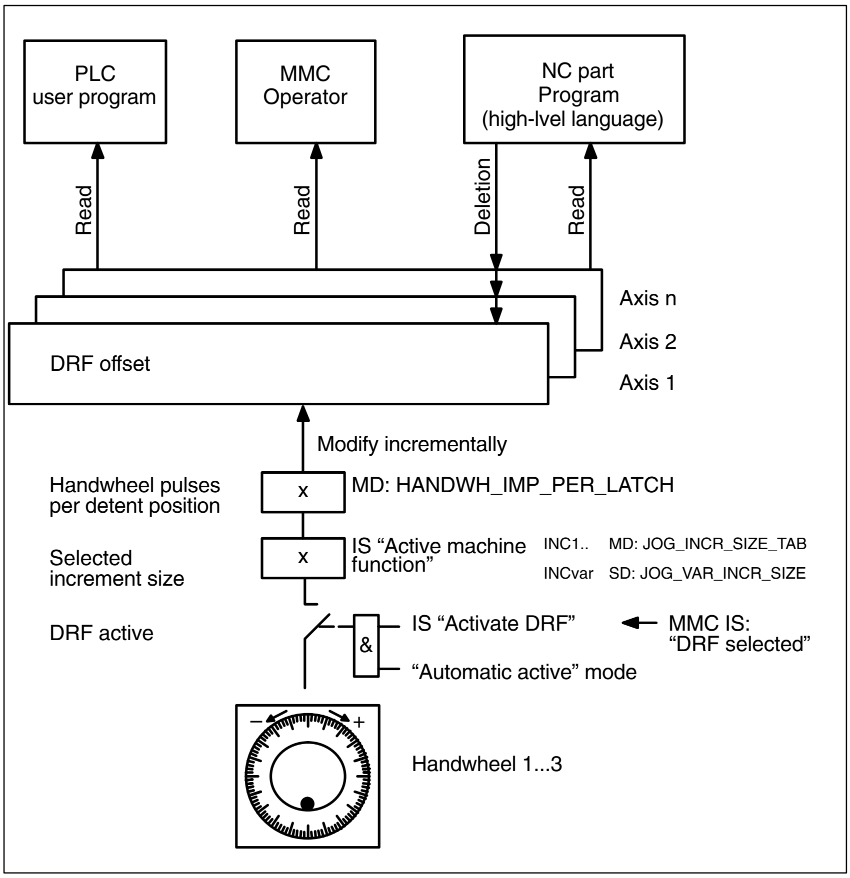



Fig. 2-7 Control of DRF offset

**Note**

If DRF offset is cleared the axis is not traversed!

### Power On

The DRF offset is deleted by a power ON.

### Reference point approach

If an axis with a DRF offset is referenced, the offset is deleted during phase 1 of the referencing operation!
It is not possible to specify a DRF offset with the handwheel while an axis is being referenced (e.g. with G74). Alarm 20053 "DRF not possible" is triggered.

### Displays

The axis position display (ACTUAL POSITION) does not change while an axis is being traversed with the handwheel in DRF. The current DRF offset can be displayed in the DRF window.

## 2.10 Start-up

---

**Note**

Before installation can begin several conditions must be fulfilled. For procedure please see:
**References:** /IAD/ "Installation and Start-Up Guide"
/IAF/ "Installation and Start-Up Guide"

---

**Machine / setting data**

The machine and/or geometry axes can be traversed manually only if specific machine/setting data have been preset. The machine and setting data that apply specifically to manual traverse are listed below. A description of these data together with their default settings is given in Chapter 4.

**JOG continuous mode**

General SD: JOG_CONT_MODE_LEVELTRIGGRD
(JOG continuously in JOG mode)

**INC and REF in jog mode**

General MD: JOG_INC_MODE_LEVELTRIGGRD
(INC and REF in JOG mode)

**Velocity**

Axial MD: JOG_VELO (JOG axis velocity)
Axial MD: JOG_VELO_RAPID (JOG rapid traverse)
General SD: JOG_SET_VELO (JOG velocity for G94)
General SD: JOG_ROT_AX_SET_VELO (JOG velocity for rotary axes)

**Revolutional feedrate**

General SD: JOG_REV_IS_ACTIVE
(revolutional feedrate active in JOG)
Axial MD: JOG_REV_VELO ((revolutional feedrate for JOG)
Axial MD: JOG_REV_VELO_RAPID
(revolutional feedrate for JOG with rapid traverse override)
General SD: JOG_REV_SET_VELO (JOG velocity for G95)

**Acceleration**

Axial MD: AX_JERK_DEFAULT (initial setting for axial jerk limitation)

| | | |
|---|---|---|
| **Incremental/ handwheel** | Axial MD: | JOG_INCR_WEIGHT (weighting of an increment for INC/handwheel) |
| | General SD: | JOG_VAR_INCR_SIZE (size of a variable increment for INC/handwheel) |
| | Axial MD: | HANDWH_VELO_OVERLAY_FACTOR (ratio JOG velocity to handwheel velocity (DRF)) |
| | General MD: | JOG_INCR_SIZE_TAB [n] (increment size for INC/handwheel) |
| | General MD: | HANDWH_IMP_PER_LATCH [n] (handwheel pulses per detent position [handwheel number]) |
| **Spindle** | General SD: | JOG_SPIND_SET_VELO (JOG velocity for spindle) |

■

# Supplementary Conditions 3

**Availability of function "Handwheel override in automatic mode"**

This function is available for:

- SINUMERIK 840D with NCU 571/572/573 with SW 2 and higher

**SINUMERIK 840Di handwheels**

Two handwheels can be connected to the SINUMERIK 840Di via the MCI Board Extension module on the SINUMERIK 840Di.
The handwheels are connected via the 25-pin cable distributor interface (X121) on the MCI Board Extension module.

# Data Descriptions (MD, SD) 4

## 4.1 General machine data

| | | | | | |
|---|---|---|---|---|---|
| **11300**<br>MD number | **JOG_INC_MODE_LEVELTRIGGRD**<br>INC and REF in jog mode | | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 1.1 | | |
| Meaning: | 1: Jog mode for JOG-INC and reference point approach<br>JOG-INC:<br>When the traverse key is pressed in the required direction (e.g. +) the axis begins to traverse the set increment. If the key is released before the increment has been completely traversed, the movement is interrupted and the axis stops.<br>If the same key is pressed again, the axis completes the remaining distance-to-go until it is 0.<br>0: Continuous mode for JOG-INC and reference point approach<br>JOG-INC:<br>When the traverse key is pressed (first rising edge) the axis traverses the whole set increment. If the same key is pressed again (second rising edge) before the axis has completed traversing the increment, the movement is aborted, i.e. not completed.<br>The differences in axis traversing characteristics between the jog and continuous modes in JOG-INC are described in detail in Section 2.3.<br>For travel behavior in reference point approach see:<br>**References:** /FB/, R1, "Reference Point Approach" | | | | |
| MD irrelevant for ...... | Continuous jogging (JOG continuous) | | | | |

| 11310<br>MD number | HANDWH_REVERSE<br>Threshold for change in handwheel direction | |
|---|---|---|
| Default setting: 2 | Minimum input limit: 0 | Maximum input limit: – |
| Changes effective after POWER ON | Protection level: 2/7 | Unit: – |
| Data type: BYTE | Applies from SW: 3.2 | |
| Meaning: | 0: No immediate movement in the opposite direction<br>>0: Immediate movement in the opposite direction if the handwheel is turned in the opposite direction by at least the number of pulses indicated | |

| 11320<br>MD number | HANDWH_IMP_PER_LATCH[n]<br>Handwheel pulses per detent position [handwheel number]: 0 ... 2 | |
|---|---|---|
| Default setting: 1 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DOUBLE | Applies from SW: 1.1 | |
| Meaning: | MD: HANDW_IMP_PER_LATCH adapts the connected handwheels to the control system. The number of pulses generated by the handwheel for each handwheel detent position is entered. The handwheel pulse weighting must be defined for each connected handwheel (1 to 3) separately.<br>When adapted to the control, each handwheel detent position has the same effect as one press of the traverse key in incremental jogging mode.<br>If a negative value is entered, the handwheel is active in the reverse direction. | |
| Related to .... | MD 31090:JOG_INCR_WEIGHT (weighting of an increment of a machine axis for INC/ manual). | |

| 11322<br>MD number | CONTOURHANDWH_IMP_PER_LATCH[n]<br>Contour handwheel pulses per detent position [handwheel number]) 0 ... 2 | |
|---|---|---|
| Default setting: 1.0 | Minimum input limit: – | Maximum input limit: – |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: DOUBLE | Applies from SW: 6.3 | |
| Meaning: | Factor for adapting to contour handwheel hardware:<br>The number of pulses output per detent position of the contour handwheel must be entered. This setting normalizes one detent position of the contour handwheel to match one key actuation in incremental JOG mode.<br>The sign reversal causes a reversal of the direction evaluation. | |
| Related to .... | MD 31090:JOG_INCR_WEIGHT (weighting of an increment of a machine axis for INC/ manual). | |

| 11330<br>MD number | JOG_INCR_SIZE_TAB[n]<br>Increment size for INC/handwheel [increment index]: 0 ... 4 | |
|---|---|---|
| Default setting: 1; 10; 100; 1000; 10000 | Minimum input limit: 0 | Maximum input limit: plus |
| Changes effective after POWER ON | Protection level: 2 | Unit:<br>Linear axis: mm<br>Rotary axis: Degrees |
| Data type: DOUBLE | Applies from SW: 1.1 | |

| | |
|---|---|
| **11330**<br>MD number | **JOG_INCR_SIZE_TAB[n]**<br>Increment size for INC/handwheel [increment index]: 0 ... 4 |
| Meaning: | In incremental jogging of handwheel jogging the number of increments to be traversed by the axis can be defined by the operator, e.g. via the operator panel. In addition to the variable increment sizes (INCvar) 5 fixed increment sizes (INC...) can also be set.<br>The increment size for each of these 5 fixed increments is defined for all axes by entering values in JOG_INCR_SIZE_TAB [n]. The default setting is INC1, INC10, INC100, INC1000 and INC10000.<br>The entered increment sizes are also active for DRF.<br>The size of the variable increment is defined in SD: JOG_VAR_INCR_SIZE. |
| Related to .... | MD 31090:JOG_INCR_WEIGHT(weighting of an increment for INC/manual)<br>IS “Active machine function: INC1; ...; INC10000” (DB21-28, DBB41 ff)<br>IS “Active machine function: INC1; ...; INC10000” (DB31-48, DBB69). |

## 4.1.1 Third handwheel via actual-value input (840D, 810D)

| | | | |
|---|---|---|---|
| **11340**<br>MD number | **ENC_HANDWEEHL_SEGMENT_NR**<br>Third handwheel: Bus segment | | |
| Default setting: 1 | Minimum input limit: 1 | Maximum input limit: 1 | |
| Changes effective after POWER ON | Protection level: 0/0 | Unit: – | |
| Data type: BYTE | Applies from SW: | 840D SW4.1<br>810D SW2.1 | |
| Meaning: | Number of bus segment via which the 3rd handwheel (encoder connection) is addressed:<br>1: Drive bus 611D<br>0, 2, 3: Reserved | | |
| Related to .... | MD 11342: ENC_HANDWEEL_MODULE_NR (third handwheel: drive no./measuring circuit no.)<br>MD 11344: ENC_HANDWHEEL_INPUT_NR (third HW: Input on module/measuring circuit card) | | |

| | | | |
|---|---|---|---|
| **11342**<br>MD number | **ENC_HANDWHEEL_MODULE_NR**<br>Third handwheel: Drive no./measuring circuit no. | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: NCU 572: 15 | |
| Changes effective after POWER ON | Protection level: 7/2 | Unit: – | |
| Data type: BYTE | Applies from SW: | 840D SW4.1<br>810D SW2.1 | |
| Meaning: | Number of the module with a segment (MD 11340: ENC_HANDWHEEL_SEGMENT_NR) via which the 3rd handwheel is addressed.<br><br>The logical drive number must be entered here for axes with 611 digital drives (see MD 13010: DRIVE_LOGIC_NR)<br>and the module number on the local bus (count from left to right)<br>(see MD 30110: CTRLOUT_MODULE_NR). | | |
| Special cases, errors, ...... | **= 0:**<br>The configuration of a 3rd handwheel is deactivated, in this case the settings in MD 11340: ENC_HANDWHEEL_SEGMENT_NR and<br>MD 11344: ENC_HANDWHEEL_INPUT_NR are irrelevant. | | |
| Related to .... | MD 13010: DRIVE_LOGIC_NR (logical drive number)<br>MD 11340: ENC_HANDWEEL_SEGMENT_NR (third handwheel: bus segment)<br>MD 11344: ENC_HANDWHEEL_INPUT_NR (third HW: Input on module/measuring circuit card) | | |

| 11344<br>MD number | ENC_HANDWHEEL_INPUT_NR<br>Third handwheel: Input on module/measuring circuit card | | |
|---|---|---|---|
| Default setting: 1 | Minimum input limit: 1 | Maximum input limit: 2 | |
| Changes effective after POWER ON | Protection level: 7/2 | Unit: – | |
| Data type: BYTE | Applies from SW: | 840D SW4.1<br>810D SW2.1 | |
| Meaning: | Number of the input on a module via which the 3rd handwheel is addressed.<br>840D: 1/2 = upper/lower actual-value input<br>810D: Always 1 | | |
| Related to .... | MD 11340: ENC_HANDWEEL_SEGMENT_NR (third handwheel: bus segment)<br>MD 11342: ENC_HANDWEEL_MODULE_NR (third handwheel: drive no./measuring circuit no.) | | |

## 4.1.2 Contour handwheel /path definition by handwheel (840D, 810D)

| 11346<br>MD number | HANDWH_TRUE_DISTANCE<br>Handwheel path/velocity values | | |
|---|---|---|---|
| Default setting: 1 | Minimum input limit: 0 | Maximum input limit: 3 | |
| Changes effective after POWER ON | Protection level: 7/2 | Unit: – | |
| Data type: BYTE | Applies from SW: | 840D SW4.1<br>810D SW2.1 | |
| Meaning: | Sets the operating characteristics for traversal with handwheel, contour handwheel or FDA=0:<br>**Value = 1:**<br>The handwheel pulses define the traversing path. No pulses are lost. Axis overtravel occurs as a result of limiting the velocity to its maximum permissible value.<br>**Value = 0:**<br>The handwheel pulses specify the axis travel velocity. As soon as the handwheel stops, the axes stop too. The axis motion is braked immediately, if no pulses are supplied by the handwheel in one IPO cycle,. As a result, the axes do not overtravel. The handwheel pulses do no define the traversing path.<br>**Value = 2**<br>The inputs from the handwheel are velocity inputs. As soon as the handwheel stops, the axes stop too. The movement is braked immediately, value = 0, but not via the shortest possible path, but to the next possible path on an imaginary grid. This grid corresponds in each case to a distance traveled by the selected axis per handwheel detent position (see MD 31090: JOG_INCR_WEIGHT and MD 11330: JOG_INCR_SIZE_TAB, MD 20620: HANDWH_GEOAX_MAX_INCR_SIZE, MD 32080: HANDWH_MAX_INCR_SIZE). The start point of the traversing motion is assumed to be the grid zero point.<br>**Value = 3**<br>The inputs from the handwheel are path inputs. If the axis needs to be decelerated prematurely owing to the settings in other MD (MD 11310: HANDWH_REVERSE = 0, MD 20624: HANDWH_CHAN_STOP_COND, MD 32084: HANDWH_STOP_COND), then unlike value = 1, the axis is not decelerated via the shortest possible path, but to the next possible point on an imaginary grid. | | |

### 4.1.3 Traversing request

| | | | | | |
|---|---|---|---|---|---|
| **17900**<br>MD number | **VDI_FUNCTION_MASK**<br>Function mask for VDI signals | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: 7 / 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: | |
| Meaning: | Bit 0 = 0:<br>The VDI signals travel command + / travel command – are output as soon as a traversing request is issued (default, response as in earlier software version).<br>Bit 0 = 1:<br>The VDI signals travel command + / travel command – are output only when the axis is actually moving. | | | | |

# 4.2 Channelspecific machine data

| | | |
|---|---|---|
| **20620**<br>MD number | **HANDWH_GEOAX_MAX_INCR_SIZE**<br>Limitation of handwheel increment for geo axes | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: – |
| Changes effective after POWER ON | Protection level: 2/7 | Unit: mm |
| Data type: DOUBLE | Applies from SW: 3.2 | |
| Meaning: | >0: Limitation of the size of the selected increment<br>MD 11330: JOG_INCR_SIZE[<Increment/VDI signal>] or<br>$SN_JOG_VAR_INCR_SIZE for geometry axes<br>0: No limitation for geometry axes | |

| | | |
|---|---|---|
| **20622**<br>MD number | **HANDWH_GEOAX_MAX_INCR_VSIZE**<br>Path velocity override | |
| Default setting: 500 | Minimum input limit: 0 | Maximum input limit: plus |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: mm/min |
| Data type: DOUBLE | Applies from SW: 3.2 | |
| Meaning: | For the velocity override of the path:<br>> 0: Limitation of size of selected increment<br>($MN_JOG_INCR_SIZE_[<Increment/VDI signal>] or<br>$SN_JOG_VAR_INCR_SIZE) / 1000*IPO sampling time<br>=0: No limitation | |

| | | |
|---|---|---|
| **20624**<br>MD number | **HANDWH_CHAN_STOP_COND**<br>Definition of operating characteristics in jogging with handwheel | |
| Default setting: 0x13FF, 0x13FF, 0x13FF, ... | Minimum input limit: 0 | Maximum input limit: 0xFFFF |
| Changes effective after POWER ON | Protection level: 2/7 | Unit: – |
| Data type: DWORD | Applies from SW: 3.2, from SW 6.4 bits 12–15 | |

<table>
<tr><td>20624<br>MD number</td><td>HANDWH_CHAN_STOP_COND<br>Definition of operating characteristics in jogging with handwheel</td></tr>
<tr><td>Meaning:</td><td>Definition of the behavior of jogging with handwheel with respect to channel-specific VDI interface signals:<br>Bit==0: Interruption or collection of the distances preset via the handwheel<br>Bit==1: Termination of the traversing motion or no collection<br>Bit allocation<br>Bit 0: Mode group stop<br>Bit 1: Mode group stop axes plus spindle<br>Bit 2: NC stop<br>Bit 3: NC stop axes plus spindles<br>Bit 4: Feedrate disable<br>Bit 5: Feedrate override<br>Bit 6: Rapid traverse override<br>Bit 7: Feed stop geometry axis<br>Bit 7=0: Interruption/collection<br>Bit 7=1: Traversing movement aborted/no collection<br>Bit 12: NC start<br><br>Setting for geometry axes:<br>Bit 8 = 0 For JOG with handwheel, the maximum possible velocity corresponds to the feedrate set in MD 32020: JOG_AX_VELO for the appropriate machine axis/axes.<br>=1 For JOG with handwheel, the maximum possible velocity corresponds to the feedrate set in MD 32000: MAX_AX_VELO for the appropriate machine axis/axes.<br>Bit 9 = 0 The override is active in JOG mode with handwheel.<br>= 1 The override is always assumed to be 100% for JOG mode with handwheel regardless of how the override switch is set;<br>Exception: The override 0% is always active.<br>Bit 14 = 0 For JOG with handwheel, the maximum possible velocity for revolutional feedrate corresponds to the feedrate in setting data SD41120: JOG_REV_SET_VELO or the feedrate in machine data MD 32050: JOG_REV_VELO_RAPID, for rapid traverse with MD 32040: JOG_REV_VELO_RAPID, for the appropriate machine axis, balanced against the spindle or rotary axis feedrate.<br>= 1 For JOG with handwheel, the maximum possible velocity for revolutional feedrate corresponds to the feedrate in machine data MD 32000: MAX_AX_VELO for the appropriate machine axis. (see also bit 6)<br>Bit 15 = 0 If the geometry axis is traversed as a transverse axis in the channel, only half the specified increment is traversed for JOG with handwheel (HANDWH_TRUE_DISTANCE == 1).<br>=1 If the geometry axis is traversed as a transverse axis in the channel, the specified increment is traversed in full for JOG with handwheel. (HANDWH_TRUE_DISTANCE == 1).<br><br>Setting for DRF for all the axes of the channel:<br>Bit 10 = 0 With DRF MD 11310: HANDWH_REVERSE is not active, i.e. the behavior is the same as with MD 11310 = 0.<br>=1 With DRF MD 11310: HANDWH_REVERSE is active.<br>Bit 13 = 0 For DRF, bits 0 to 3 and bit 12 == 0 / bit == 1 are active, see above.<br>=1 For DRF, bits 0 to 3 and bit 12 are not, active, i.e. DRF motion is not interrupted by a Stop and even if interrupted in automatic mode (brought about by NC Stop), DRF motion can occur.<br>Note: If an axis is stopped by an alarm and an alarm of this type is pending, no DRF motion can occur.<br><br>Setting for the contour handwheel<br>Bit 11 = 0 When the contour handwheel is deactivated, program execution is automatically continued.<br>Bit 11 = 1 When the contour handwheel is deactivated, an NC stop is automatically initiated. Only after input of NC-START, can the program execution be continued.</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| **21106**<br>MD number | **CART_JOG_SYSTEM**<br>Coordinate system for cart. JOG | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 7 | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 6.3 | |
| Meaning: | This machine data has two different meanings.<br><br>On the one hand, it is used to activate the function Cartesian manual travel.<br><br>On the other, it can be used to specify the referencing systems for switchover. The meanings of the individual bits is defined as follows:<br><br>Bit 0: Basic Coordinate System<br>Bit 1: Workpiece coordinate system<br>Bit 2: Tool coordinate system | | | | |

## 4.3 Axis/spindle-specific machine data

| | | | | | |
|---|---|---|---|---|---|
| **31090**<br>MD number | **JOG_INCR_WEIGHT**<br>Evaluation of an increment for INC/handwheel | | | | |
| Default setting: 0.001 | | Minimum input limit: *** | | Maximum input limit: *** | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit:<br>Linear axis: mm<br>Rotary axis: Degrees |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | |
| Meaning: | The **path of an increment** which applies when an axis is traversed with the JOG keys in incremental mode or with the handwheel is defined in this MD.<br>The path covered by the axis on each increment each time the direction key is pressed or for each handwheel position is defined by the following parameters:<br>• MD 31090: JOG_INCR_WEIGHT (weighting of an increment of a machine axis for INC/handwheel)<br>• Selected increment size (INC1, ..., INCvar)<br>The possible increment stages are defined globally for all axes in MD 11330: JOG_INCR_SIZE_TAB [n] or in SD 41010: JOG_VAR_INCR_SIZE.<br>Entering a negative value reverses the direction of the traverse keys and the handwheel rotation.<br><br>**SW 5 and higher:**<br>Default settings:<br>JOG_INCR_WEIGHT[0]=0.001 mm (valid in metric measuring system)<br>JOG_INCR_WEIGHT[1]=0.00254 mm (valid in inch measuring system and corresponds to 0.0001 inch) | | | | |
| MD irrelevant for ...... | Operating modes AUTOMATIC and MDA | | | | |
| Related to .... | MD 11330: JOG_INCR_SIZE_TAB<br>SD 41010: JOG_VAR_INCR_SIZE | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **32010**<br>MD number | **JOG_VELO_RAPID**<br>Rapid traverse in JOG mode | | | | |
| Default setting: 10000 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit:<br>Linear axis: mm/min<br>Rotary axis: rpm |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | |
| Meaning: | The axis velocity entered here applies when the rapid traverse override key is operated in JOG mode and when the axial feedrate override switch is set to 100%.<br>The value entered must not exceed the maximum permissible axis velocity in the machine data (MD 32000: MAX_AX_VELO).<br>This machine data will not be used for the programmed rapid traverse G00. | | | | |
| MD irrelevant for ...... | Operating modes AUTOMATIC and MDA | | | | |
| Related to .... | MD 32000: MAX_AX_VELO (maximum axis velocity)<br>MD 32040: JOG_REV_VELO_RAPID (JOG revolutional feedrate with rapid traverse)<br>IS “Rapid traverse override” (DB21-28, DBX12.5 ff)<br>IS “Feedrate override” (DB21-28, DBB4) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **32020**<br>MD number | **JOG_VELO**<br>Axis velocity in JOG mode | | | | |
| Default setting: 2000 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit:<br>Linear axis: mm/min<br>Rotary axis: rpm |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | |
| Meaning: | The value entered is the velocity traversed in JOG mode when the axial feedrate override switch is on position 100%.<br>This velocity is only used when general setting data<br>SD 41110: JOG_SET_VELO = 0 is set for linear axes and linear feedrate<br>(SD 41100: JOG_REV_IS_ACTIVE = 0) or<br>SD 41130: JOG_ROT_AX_SET_VELO = 0 for rotary axes.<br>If this is the case, the axis velocity is active for<br>– continuous jogging<br>– incremental jogging (INC1, ... INCvar)<br>– handwheel jogging<br>The value entered must not exceed the maximum permissible axis velocity in the machine data (MD 32000: MAX_AX_VELO).<br>If DRF is active, the axis velocity for JOG must be reduced with<br>MD 32090: HANDWH_VELO_OVERLAY_FACTOR.<br><br>Spindles in JOG mode:<br>This machine data can also be used to define the JOG mode velocity for specific spindles (if SD 41200: JOG_SPIND_SET_VELO = 0). However, the velocity can be modified with the spindle speed override switch. | | | | |
| Application example(s) | If different velocities/speeds have to be set for the individual axes/spindles traversing in JOG mode, this can be done for specific axes in this MD. SD: JOG_SET_VELO must be set to 0! | | | | |
| Related to .... | MD 32000: MAX_AX_VELO (maximum axis velocity)<br>MD 32050: JOG_REV_VELO (revolutional feedrate for JOG)<br>MD 32090: HANDWH_VELO_OVERLAY_FACTOR (ratio JOG velocity to handwheel velocity (DRF))<br>SD 41110: JOG_SET_VELO (JOG velocity for G94)<br>SD 41130: JOG_ROT_AX_SET_VELO (JOG velocity for rotary axes)<br>IS “Feedrate override” (DB21-28, DBB4) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **32040**<br>MD number | **JOG_REV_VELO_RAPID**<br>Revolutional feedrate in JOG mode with rapid traverse override | | | | |
| Default setting: 2,5 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: mm/rev. |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | |
| Meaning: | The value entered in this MD defines the revolutional feedrate of the axis in JOG mode with rapid traverse override referred to the revolutions of the master spindle.<br>This feedrate is active when SD 41100: JOG_REV_IS_ACTIVE = 1.<br>(revolutional feedrate active in JOG) | | | | |
| MD irrelevant for ...... | SD 41100 JOG_REV_IS_ACTIVE = “0” | | | | |
| Related to .... | SD 41100: JOG_REV_IS_ACTIVE (revolutional feedrate for JOG active)<br>MD 32050: JOG_REV_VELO (revolutional feedrate for JOG mode) | | | | |

| | | | |
|---|---|---|---|
| **32050**<br>MD number | **JOG_REV_VELO**<br>Revolutional feedrate in JOG mode | | |
| Default setting: 0,5 | Minimum input limit: 0 | | Maximum input limit: plus |
| Changes effective after POWER ON | Protection level: 2 | | Unit: mm/rev. |
| Data type: DOUBLE | | Applies from SW: 1.1 | |
| Meaning: | The value entered in this MD defines the revolutional feedrate of the axis in JOG mode referred to the revolutions of the master spindle.<br>This feedrate value is active if the revolutional feedrate setting data for JOG is active, SD 41100: JOG_REV_IS_ACTIVE = 1. | | |
| MD irrelevant for ...... | Linear feedrate; i.e. SD 41100: JOG_REV_IS_ACTIVE = 0 | | |
| Related to .... | SD 41100: JOG_REV_IS_ACTIVE (revolutional feedrate for JOG active)<br>MD 32040: JOG_REV_VELO_RAPID (JOG revolutional feedrate with rapid traverse) | | |

| | | | |
|---|---|---|---|
| **32080**<br>MD number | **HANDWH_MAX_INCR_SIZE**<br>Limitation of selected increment | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: – |
| Changes effective after Reset | Protection level: 2/7 | | Unit: |
| Data type: DOUBLE | | Applies from SW: 3.2 | |
| Meaning: | >0: Limitation of the size of the selected increment<br>MD 11330: JOG_INCR_SIZE[<Increment/VDI signal>] or<br>SD 41010: JOG_VAR_INCR_SIZE for the associated machine axis<br>0: No limitation | | |

| | | | |
|---|---|---|---|
| **32082**<br>MD number | **HANDWH_MAX_INCR_VELO_SIZE**<br>Limitation of selected increment for velocity override | | |
| Default setting: 500 | Minimum input limit: 0 | | Maximum input limit: plus |
| Changes effective after Reset | Protection level: 2/7 | | Unit: mm/min |
| Data type: DOUBLE | | Applies from SW: 3.2 | |
| Meaning: | For the velocity override of positioning axes:<br>>0: Limitation of the size of the selected increment<br>MD 11330: JOG_INCR_SIZE[<Increment/VDI signal>] or<br>SD 41010: JOG_VAR_INCR_SIZE for the associated machine axis<br>0: No limitation | | |

| | | | | | |
|---|---|---|---|---|---|
| **32084**<br>MD number | **HANDWH_STOP_COND**<br>Effect of the VDI signals on the handwheel | | | | |
| Default setting: 0xFF | | Minimum input limit: 0 | | Maximum input limit: 0x7FF | |
| Changes effective after RESET | | | Protection level: 2/7 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 3.2 | | |
| Meaning: | Definition of the behavior of jogging with handwheel in response to axis–specific VDI interface signals:<br>Bit==0: Interruption or collection of the distances preset via the handwheel<br>Bit==1: Abort of the traversing movement or no collection<br>Bit allocation:<br>Bit 0: Feedrate override<br>Bit 1: Spindle speed override<br>Bit 2: Feedrate stop/spindle stop<br>Bit 3: Clamping in progress (== 0 no effect)<br>Bit 4: Servo enable<br>Bit 5: Pulse enable<br><br>For machine axis<br>Bit 6 == 0 For JOG with handwheel, the maximum possible velocity corresponds to the feedrate set in MD 32020:JOG_VELO for the appropriate machine axis.<br>==1 For JOG with handwheel, the maximum possible velocity corresponds to the feedrate set in MD 32000:MAX_AX_VELO for the appropriate machine axis.<br><br>Bit 7 == 0 The override is active in JOG mode with handwheel.<br>== 1 The override is always assumed to be 100% for JOG mode with handwheel regardless of how the override switch is set;<br>Exception: The override 0% is always active.<br>**SW 6.3 and higher**<br>Bit 8 == 0 The override is active for JOG mode with DRF offset.<br>== 1 The override is always assumed to be 100% for JOG mode with handwheel regardless of how the override switch is set.<br>Exception: The override 0% is always active.<br><br>Bit 9 == 0 For JOG with handwheel, the maximum possible velocity for revolutional feedrate (G95) corresponds to the feedrate in setting data in SD 41120: JOG_REV_SET_VELO or the feedrate in machine data MD 32050: JOG_REV_VELO, or for rapid traverse with MD 32040: JOG_REV_VELO_RAPID, for the appropriate machine axis, balanced against the spindle or rotary axis feedrate.<br><br>= 1 For JOG with handwheel, the maximum possible velocity for revolutional feedrate corresponds to the feedrate in machine data MD 32000: MAX_AX_VELO for the appropriate machine axis. (see also bit 6)<br>**SW 6.4 and higher**<br>Bit 10 == 0 $AA_OVR is not active for overlaid movements.<br>== 1 The $AA_OVR override, which can be set via synchronized actions, is active for overlaid movements (DRF offset, $AA_OFF, zero offset external, online tool offset). | | | | | |
| Related to .... | MD 20624: HANDWH_CHAN_STOP_COND (definition of the behavior of jogging with handwheel) | | | | |

| | | | |
|---|---|---|---|
| **32090**<br>MD number | **HANDWH_VELO_OVERLAY_FACTOR**<br>Ratio JOG velocity to handwheel velocity (DRF) | | |
| Default setting: 0,5 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after NEW_CONF | Protection level: 2 | Unit: – | |
| Data type: DOUBLE | Applies from SW: 1.1 | | |
| Meaning: | The velocity active with the handwheel in DRF can be reduced in relation to the JOG velocity with this machine data.<br>The following applies for the velocity active with DRF:<br>$v_{DRF}$ = SD:JOG_SET_VELO * MD:HANDWH_VELO_OVERLAY_FACTOR<br>or when SD:JOG_SET_VELO = 0:<br>$v_{DRF}$ = MD:JOG_VELO * MD:HANDWH_VELO_OVERLAY_FACTOR<br>The velocity setting in SD: JOG_ROT_AX_SET_VELO applies for DRF on rotary axes instead of the value in SD: JOG_ROT_SET_VELO. | | |
| MD irrelevant for ...... | JOG handwheel | | |
| Related to .... | MD: JOG_VELO (JOG axis velocity)<br>SD: JOG_SET_VELO (JOG velocity for G94)<br>SD: JOG_AX_SET_VELO (JOG velocity for rotary axes) | | |

# 4.4 General setting data

| | | | |
|---|---|---|---|
| **41010**<br>SD number | **JOG_VAR_INCR_SIZE**<br>Size of variable increment for INC/handwheel | | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** | |
| Changes effective immediately | Protection level: MMCMD 9220 | Unit: mm or degrees | |
| Data type: DOUBLE | Applies from SW: 1.1 | | |
| Meaning: | This setting data defines the number of increments when variable increment (INCvar) is selected. This increment size is traversed by the axis in JOG mode whenever the traverse key is pressed or the handwheel is turned one detent position and variable increment is selected (PLC interface signal “Active machine function: INC variable” interface signal for machine or geometry axes is set to 1).<br>The defined increment size also applies to DRF.<br>**Note:** Please note that the increment size is active for incremental jogging **and** handwheel jogging. So if a large increment value is entered and the handwheel is turned the axis might cover a large distance (depends on setting in MD: JOG_INCR_WEIGHT). | | |
| SD irrelevant for ...... | JOG continuous | | |
| Related to .... | IS “Active machine function: INC variable” (DB21-28, DBX41.5 ff)<br>or IS “Active machine function; INC variable” (DB31-48, DBX 69.5)<br>MD: JOG_INCR_WEIGHT (weighting of an increment for INC/handwheel) | | |

| | | | | |
|---|---|---|---|---|
| **41050** SD number | **JOG_CONT_MODE_LEVELTRIGGRD** Continuous JOG in jog mode | | | |
| Default setting: 1 | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective immediately | | Protection level: MMCMD 9220 | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 1.1 | |
| Meaning: | 1: Jog mode for JOG continuous<br>In jog mode (initial setting) the axis traverses for as long as the traverse key is held down and an axis limitation has not been reached. When the traverse key is released the axis is decelerated to zero speed and the movement comes to an end.<br>0: Continuous mode for JOG continuous<br>In continuous mode the traverse movement is started with the first rising edge of the traverse key and continues to move when the key is released. The axis can be stopped again by pressing the traverse key again (second rising edge).<br>The differences in axis traversing characteristics between the jog and continuous modes in JOG are described in detail in Section 2.1. | | | |
| SD irrelevant for ...... | Incremental jogging (JOG INC)<br>Reference point approach (JOG REF) | | | |

| | | | | |
|---|---|---|---|---|
| **41100** SD number | **JOG_REV_IS_ACTIVE** Revolutional feedrate for JOG active | | | |
| Default setting: 1 | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective immediately | | Protection level: MMCMD 9220 | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 1.1 | |
| Meaning: | 1: The axis (machine or geometry axis) is traversed in JOG mode at revolutional feedrate (G95) referred to the revolutions of the main spindle.<br>The revolutional feedrate can be set as follows:<br>– With global SD: JOG_REV_SET_VELO (only active when SD is not equal to 0)<br>– With axial MD: JOG_REV_VELO<br>– With rapid traverse override with axial MD: JOG_REV_VELO_RAPID<br>0: The axis is traversed in JOG mode at linear feedrate (G94).<br>The linear feedrate value can be set as follows:<br>– With global SD: JOG_SET_VELO (only active when SD is not equal to 0)<br>– With axial MD: JOG_VELO<br>– With rapid traverse override with axial MD: JOG_VELO_RAPID | | | |
| SD irrelevant for ...... | Operating modes AUTOMATIC and MDA | | | |
| Related to .... | SD: JOG_REV_SET_VELO (JOG velocity for G95)<br>MD: JOG_REV_VELO (revolutional feedrate for JOG mode)<br>MD: JOG_REV_VELO_RAPID (JOG revolutional feedrate with rapid traverse)<br>SD: JOG_SET_VELO (JOG velocity for G94)<br>MD: JOG_VELO (JOG axis velocity)<br>MD: JOG_VELO_RAPID (JOG rapid traverse) | | | |

| **41110**<br>SD number | **JOG_SET_VELO**<br>JOG velocity for linear axes (for G94) |
|---|---|
| Default setting: 0 \| Minimum input limit: 0 \| Maximum input limit: plus | |
| Changes effective immediately \| Protection level: MMCMD 9220 \| Unit: mm/min or rpm | |
| Data type: DOUBLE \| Applies from SW: 1.1 | |
| Meaning: | Value not equal to zero:<br>The velocity value entered applies to linear axes traversed in JOG mode if linear feedrate (G94) is active for the relevant axis (MD: JOG_REV_IS_ACTIVE = 0).<br>The axis velocity is active for:<br>– Continuous jogging<br>– Incremental jogging (INC1, ... INCvar)<br>– Handwheel jogging<br>The value entered is valid for all linear axes and must not exceed the maximum permissible axis velocity (MD: MAX_AX_VELO).<br>With DRF: If DRF is active, the velocity set in SD:JOG_SET_VELO must be reduced with MD: HANDWH_VELO_OVERLAY_FACTOR.<br>Value = 0:<br>If 0 has been entered in the setting data, the active linear feedrate in JOG mode is MD: JOG_VELO “JOG axis velocity”. Each axis can be given its own JOG velocity with this MD (axial MD). |
| SD irrelevant for ...... | – For linear axes if SD: JOG_REV_IS_ACTIVE = 1<br>– For rotary axes (SD: JOG_ROT_AX_SET_VELO) applies |
| Application example(s) | The operator can define a JOG velocity for a particular application. |
| Related to .... | SD: JOG_REV_IS_ACTIVE (revolutional feedrate for JOG active)<br>Axial MD: JOG_VELO (JOG axis velocity)<br>Axial MD: MAX_AX_VELO (maximum axis velocity)<br>Axial MD: HANDWH_VELO_OVERLAY_FACTOR (ratio JOG velocity to handwheel velocity (DRF))<br>SD: JOG_ROT_AX_SET_VELO (JOG velocity for rotary axes) |

| **41120**<br>SD number | **JOG_REV_SET_VELO**<br>JOG velocity (for G95) |
|---|---|
| Default setting: 0 \| Minimum input limit: 0 \| Maximum input limit: plus | |
| Changes effective immediately \| Protection level: MMCMD 9220 \| Unit: mm/rev. | |
| Data type: DOUBLE \| Applies from SW: 1.1 | |
| Meaning: | Value not equal to zero:<br>The velocity value entered applies to axes traversed in JOG mode if revolutional feedrate (G95) is active for the relevant axis (MD: JOG_REV_IS_ACTIVE = 1).<br>The axis velocity is active for:<br>– Continuous jogging<br>– Incremental jogging (INC1, ... INCvar)<br>– handwheel jogging<br>The value entered is valid for all axes and must not exceed the maximum permissible axis velocity (MD: MAX_AX_VELO).<br>Value = 0:<br>If 0 has been entered in the setting data, the active revolutional feedrate in JOG mode is MD: JOG_REV_VELO “revolutional feedrate with JOG”.<br>Each axis can be given its own revolutional feedrate with this MD (axial MD). |
| SD irrelevant for ...... | – For axes if SD: JOG_REV_IS_ACTIVE = 0 |
| Application example(s) | The operator can define a JOG velocity for a particular application. |
| Related to .... | Axial SD: JOG_REV_IS_ACTIVE (revolutional feedrate for JOG active)<br>Axial MD: JOG_REV_VELO (revolutional feedrate for JOG mode)<br>Axial MD: MAX_AX_VELO (maximum axis velocity) |

| **41130**<br>SD number | **JOG_ROT_AX_SET_VELO**<br>JOG velocity for rotary axes | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective immediately | Protection level: MMCMD 9220 | Unit: rpm | |
| Data type: DOUBLE | Applies from SW: 2.1 | | |
| Meaning: | Value not equal to zero:<br>The velocity entered applies to rotary axes in JOG mode (in continuous mode, in incremental mode, in jogging with handwheel).<br>The value entered is common to all rotary axes and must not exceed the maximum permissible axis velocity (MD: MAX_AX_VELO).<br><br>With DRF, the velocity set with SD: JOG_ROT_AX_SET_VELO must be reduced with the MD: HANDWH_VELO_OVERLAY_FACTOR.<br><br>Value = 0:<br>If a value of 0 is entered in the setting data, then the velocity in JOG mode for rotary axes is the setting in axial MD: JOG_VELO (JOG axis velocity). In this way, it is possible to define a separate JOG velocity for every axis. | | |
| Application example(s) | The operator can define a JOG velocity for a particular application. | | |
| Related to .... | MD: JOG_VELO (JOG axis velocity)<br>MD: MAX_AX_VELO (maximum axis velocity)<br>MD: HANDWH_VELO_OVERLAY_FACTOR (ratio JOG velocity to handwheel velocity (DRF) | | |

| **41200**<br>SD number | **JOG_SPIND_SET_VELO**<br>JOG velocity for spindles | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective immediately | Protection level: MMCMD 9220 | Unit: rpm | |
| Data type: DOUBLE | Applies from SW: 1.1 | | |
| Meaning: | Value not equal to zero:<br>The velocity entered applies to spindles in JOG mode if they are traversed manually using the “traversing keys plus and minus”.<br>The velocity setting is active in:<br>– Continuous jogging<br>– Incremental jogging (INC1, ... INCvar)<br>– Handwheel jogging<br>The value entered is valid for all spindles and must not exceed the maximum permissible velocity (MD: MAX_AX_VELO).<br>Value = 0:<br>If 0 has been entered in the setting data, the active JOG velocity is MD: JOG_VELO (JOG axis velocity). Each axis can be given its own JOG velocity with this MD (axial MD).<br>When the spindle is traversed in JOG mode, the maximum velocity of the active gear stage (MD: GEAR_STEP_VELO_LIMIT) is taken into account. | | |
| SD irrelevant for ...... | Axes | | |
| Application example(s) | The operator can define a JOG velocity for the spindles for a particular application. | | |
| Related to .... | Axial MD: JOG_VELO (JOG axis velocity)<br>MD: GEAR_STEP_MAX_VELO_LIMIT (maximum velocity of gear stages) | | |
| References | /FB/, S1, “Spindles” | | |

## 4.5 Channelspecific setting data

| | | | | | |
|---|---|---|---|---|---|
| **42650** | **CART_JOG_MODE** | | | | |
| MD number | Coordinate system for cart. manual travel | | | | |
| Default setting: 0x0 | | Minimum input limit: 0 | | Maximum input limit: 0x0404 | |
| Changes effective IMMEDIATELY | | | Protection level: 7 / 7 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 6.3 | |
| Meaning: | This allows the reference coordinate system to be set for Cartesian manual travel.<br>Bits 0 to 7 are provided for selecting the coordinate system for the translation, Bits 8 to 15 for selecting the reference system for the orientation.<br>If no bit is set, or only one bit either for the translation or for the orientation, the Cartesian manual travel is not active.<br>This means that one bit must always be set for the translation and one bit set for the orientation. If more than one bit is set for the translation or for the orientation, the Cartesian manual travel is not active either.<br><br>The meaning of the individual bits is defined as follows:<br>Bit 0: Translation in the Basic Coordinate System<br>Bit 1: Translation in the Workpiece Coordinate System<br>Bit 2: Translation in the Tool Coordinate System<br>Bit 3: Reserved<br>Bit 4: Reserved<br>Bit 5: Reserved<br>Bit 6: Reserved<br>Bit 7: Reserved<br>Bit 8: Orientation in the Basic Coordinate System<br>Bit 9: Orientation in the Workpiece Coordinate System<br>Bit 10: Orientation in the Tool Coordinate System<br>Bit 11: Reserved<br>Bit 12: Reserved<br>Bit 13: Reserved<br>Bit 14: Reserved<br>Bit 15: Reserved | | | | |
| MD irrelevant for ...... | | | | | |
| Figure ????.??? | | | | | |
| Application example(s) | | | | | |
| Special cases, errors, ...... | | | | | |
| Related to .... | | | | | |
| References | | | | | |

■

# Notes

# Signal Descriptions 5

## 5.1 General signals

### 5.1.1 Signals from NC

| **DB10**<br>**DBB97, 98, 99**<br>Data Block | **Channel number of geometry axis for**<br>**Handwheel 1, 2, 3**<br>Signal(s) from NC (MMC –> PLC) |
|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic — Signal(s) valid from SW: 2.1 |
| Significance of signal | 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.<br>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").<br>The following codes apply to the channel number:<br>Bit 7 / 6 / 5 / 4 / 3 / 2 / 1 / Bit 0 — Channel number<br>0 0 0 0 0 0 0 0 — –<br>0 0 0 0 0 0 0 1 — 1<br>0 0 0 0 0 0 1 0 — 2<br>With machine axes (IS "Machine axis" = 1), the IS "Channel number geometry axis for handwheel 1, 2, 3" has no meaning.<br>For further information, see IS "Axis number for handwheel 1, 2, 3". |
| Related to .... | IS "Axis number of handwheel 1, 2, 3" (DB10, DBB100 ff)<br>IS "Handwheel selected" (DB10, DBX100.6 ff)<br>IS "Machine axis" (DB10, DBX100.7 ff)<br>IS "Activate handwheel" (DB21, ... DBX12.0 – 12.2 ff) |
| Application example(s) | If DB10, DBB97 = 2, then handwheel 1 is assigned to channel 2. |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBB100, 101, 102,**<br>**Bits 0 –4**<br>Data Block | **Axis number for handwheel 1, 2 or 3**<br><br>Signal(s) from NC (MMC –> PLC) | | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 | |
| 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).<br>The basic PLC program provides the number of the axis plus the information 'machine axis or geometry axis' (IS "machine axis") as MMC interface signals.<br>The basic PLC program sets the interface signal "Activate handwheel" for the defined axis. Depending on the setting in the MMC interface signal "machine axis", either the interface for the geometry axis or for the machine axis is used.<br>The following must be noted when assigning the axis designation to the axis number:<br>IS "Machine axis" = 1; i.e. machine axis:<br>The assignment is made via MD: AXCONF_MACHAX_NAME_TAB[n]<br>(machine axis name).<br>IS "Machine axis" = 0; i.e. geometry axis:<br>The assignment is made via MD: AXCONF_GEOAX_NAME_TAB[n]<br>(geometry axis name in channel). IS "Channel number geometry axis handwheel n" defines the channel assigned to the handwheel.<br>For following codes are used for the axis number: | | |

| Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Axis number |
|---|---|---|---|---|---|
| 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 |

| | |
|---|---|
| Related to .... | IS "Channel number geometry axis handwheel n" (DB10, DBX97 ff)<br>IS "Handwheel selected" (DB10, DBX100.6 ff)<br>IS "Machine axis" (DB10, DBX100.7 ff)<br>IS "Activate handwheel" (DB21, ... DBX12.0 to DBX12.2 ff)<br>IS "Activate handwheel" (DB31, ... DBX4.0 to DBX4.2)<br>MD: AXCONF_MACHAX_NAME_TAB [n] (machine axis name)<br>MD: AXCONF_GEOAX_NAME_TAB[n] (name of the geometry axis in the channel) |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBX100.6; 101.6; 102.6**<br>Data Block | **Handwheel selected (for handwheel 1, 2 or 3)**<br>Signal(s) from NC (MMC –> PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 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 MMC interface.<br>This means that the interface signal "Activate handwheel" is set to '1' for the defined axis by the basic PLC program.<br>The associated axis is also displayed on the MMC interface (IS: "Axis number" and IS "Machine axis").<br>As soon as the handwheel is active, the axis can be traversed in JOG mode with the handwheel (IS "Handwheel active"). | | |
| Signal state 0 or signal transition 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 MMC interface.<br>Now the interface signal "Activate handwheel" can be reset for the defined axis by the basic PLC program. | | |
| Related to .... | IS "Axis number" (DB10, DBB100 ff)<br>IS "Machine axis" (DB10, DBX100.7 ff)<br>IS "Activate handwheel" (DB21, ... DBX12.0 – DBX12.2 ff)<br>IS "Handwheel active" (DB21, ... DBX40.0 – DBX40.2 ff)<br>IS "Activate handwheel" (DB31, ... DBX4.0 – DBX4.2)<br>IS "Channel number geometry axis for handwheel 1, 2 or 3" (DB10, DBB97 ff) | | |

| | | | |
|---|---|---|---|
| **DB10**<br>**DBX100.7; 101.7; 102.7**<br>Data Block | **Machine axis (for handwheel 1, 2 or 3)**<br>Signal(s) from NC (MMC –> PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 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.<br>For further information see IS "Axis number". | | |
| Signal state 0 or signal transition 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.<br>For further information see IS "Axis number". | | |
| Related to .... | IS "Axis number" (DB10, DBB100 ff)<br>IS "Handwheel selected" (DB10, DBX100.6 ff)<br>IS "Channel number geometry axis for handwheel 1, 2 or 3" (DB10, DBB97 ff) | | |

# 5.2 Channel-specific signals

## 5.2.1 Overview of signals to channel (to NCK)

| DB 21, ... | Signals to channel | | | | | | |
|---|---|---|---|---|---|---|---|
| DBB | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | | Activate dry run feed | Activate M01 | Activate single block | Activate DRF | Activate traverse forwards | Activate traverse backwards | Execution from external source |
| | Geometry axis 1 | | | | | | | |
| 12 | Traversing keys + | Traversing keys – | Rapid traverse override | Traversing key lock | Feed stop | Activate handwheel 3 | Activate handwheel 2 | Activate handwheel 1 |
| | Geometry axis 1<br>Machine function | | | | | | | |
| 13 | | Continuous | Variable INC | 10000 INC | 1000 INC | 100 INC | 10 INC | 1 INC |
| | Geometry axis 2 | | | | | | | |
| 16 | Traversing keys + | Traversing keys – | Rapid traverse override | Traversing key lock | Feed stop | Activate handwheel 3 | Activate handwheel 2 | Activate handwheel 1 |
| | Geometry axis 2<br>Machine function | | | | | | | |
| 17 | | Continuous | Variable INC | 10000 INC | 1000 INC | 100 INC | 10 INC | 1 INC |
| | Geometry axis 3 | | | | | | | |
| 20 | Traversing keys + | Traversing keys – | Rapid traverse override | Traversing key lock | Feed stop | Activate handwheel 3 | Activate handwheel 2 | Activate handwheel 1 |
| | Geometry axis 3<br>Machine function | | | | | | | |
| 21 | | Continuous | Variable INC | 10000 INC | 1000 INC | 100 INC | 10 INC | 1 INC |

## 5.2.2 Overview of signals to channel (to NCK)

| | | |
|---|---|---|
| **DB21, ...**<br>**DBX0.3**<br>Data Block | **Activate DRF**<br>Signal(s) to channel (PLC –> NCK) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The function DRF is selected.<br>The function can either be selected directly from the PLC user program or from the operator panel front via MMC interface signal “DRF selected”. This MMC interface signal is either converted by the basic PLC program or the PLC user program to interface signal “Activate DRF”.<br>As soon as the function DRF is active, DRF offset can be modified in operating modes AUTOMATIC or MDA. | |
| Signal state 0 or signal transition 1 ––> 0 | The function is not selected. | |
| Signal irrelevant for ...... | JOG mode | |
| Application example(s) | The DRF function can be enabled specifically by the PLC user program with IS “Activate DRF”. | |
| Related to .... | IS “DRF selected” (DB21, ... DBX24.3) | |

| | | |
|---|---|---|
| **DB21, ...**<br>**DBB12; 16; 20 Bits 0-2**<br>Data Block | **Activate handwheel (1 to 3) for geometry axis (1,2,3)**<br>Signal(s) to channel (PLC –> NCK) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | These machine data determine whether this geometry axis is assigned to handwheel 1, 2, 3 or no handwheel.<br>Only one handwheel can be assigned to an axis at any one time.<br>If several “Activate handwheel” interface signals are set, priority “Handwheel 1” before “Handwheel 2” before “Handwheel 3” applies.<br><br>**Note:** Three geometry axes can be traversed simultaneously with handwheels 1 to 3! | |
| Signal state 0 or signal transition 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 geometry axis. | |
| Related to .... | IS “Handwheel active” for geometry axis (DB21, ... DBX40.7 or DBX40.6 ff) | |

| | | |
|---|---|---|
| **DB21, ...**<br>**DBX12.4; 16.4; 20.4**<br>Data Block | **Traverse key disable for geometry axis (1,2,3)**<br>Signal(s) to channel (PLC –> NCK) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The traverse keys plus and minus have no effect on the geometry axes in question. It is thus not possible to traverse the geometry axis in JOG with the traverse keys on the machine control panel.<br>If the traverse key disable is activated during a traverse movement, the geometry axis is stopped. | |
| Signal state 0 | Traverse keys plus and minus are enabled. | |
| Application example(s) | It is thus possible, depending on the operating mode, to disable manual traverse of the geometry axis in JOG mode with the traverse keys from the PLC user program. | |
| Related to .... | IS “Traverse key plus” and “Traverse key minus” for geometry axis (DB21, ... DBX12.7 or DBX12.6 ff) | |

| | | | |
|---|---|---|---|
| **DB21, ...**<br>**DBX12.5; 16.5; 20.5**<br>Data Block | **Rapid traverse override for geometry axis (1,2,3)**<br>Signal(s) to channel (PLC –> NCK) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | If interface signal “Rapid traverse override” is set together with “Traverse key plus” and “Traverse key plus”, the geometry key in question traverses at rapid traverse.<br>The rapid traverse feedrate is defined in machine data JOG_VELO_RAPID.<br>Rapid traverse override is active in the following JOG modes:<br>– Continuous jogging<br>– Incremental jogging<br>If rapid traverse override is active, the velocity can be modified with the rapid traverse override switch. | | |
| Signal state 0 or signal transition 1 ––> 0 | The geometry axis traverses at the defined JOG velocity (SD: JOG_SET_VELO or MD: JOG_VELO). | | |
| Signal irrelevant for ...... | – Operating modes AUTOMATIC and MDA<br>– Reference point approach (JOG mode) | | |
| Related to .... | IS “Traverse key plus” and “Traverse key minus” for geometry axis (DB21, ... DBX12.7 or DBX12.6 ff) | | |
| References | /FB/, V1, “Feeds” | | |

| **DB21, ...**<br>**DBX12, 16, 20**<br>**Bits 7, 6**<br>Data Block | **Plus and minus traverse keys for geometry axis (1,2,3)**<br><br>Signal(s) to channel (PLC –> NCK) |
|---|---|
| Edge evaluation: yes | Signal(s) updated: Cyclic — Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | The selected geometry axis can be traversed in both directions in JOG mode with the traverse keys plus and minus.<br>Depending on the active machine function and the setting "Jog or continuous mode" (SD: JOG_CONT_MODE_LEVELTRIGGRD for JOG continuous and MD: JOG_INC_MODE_LEVELTRIGGRD for JOG INC), the signal transition will cause different reactions.<br>Case 1: **Continuous jogging with jog mode**<br>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).<br>Case 2: **Continuous jogging with continuous mode**<br>On the first signal edge change from 0 → 1 the geometry axis starts to traverse in the relevant direction. This traversing movement still continues when the edge changes from 1 → 0. A new signal edge change from 0 → 1 (same traversing direction!) stops the traversing movement.<br>Case 3: **Incremental jogging with jog mode**<br>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.<br>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 the increment has been completely traversed.<br>Case 4: **Incremental jogging with continuous mode**<br>On the first signal edge change from 0→ 1 the geometry axis starts to traverse at the set increment. If the same traversing signal is applied and the edge changes from 0 → again before the geometry axis has traversed the increment, the traverse movement is aborted.<br>The increment is not traversed to the end.<br>If both traverse signals (plus and minus) are set at the same time, no movement occurs, or any current movement is aborted!<br>The effect of the traverse keys can be disabled for every geometry axis individually with the PLC interface signal "Traverse key disable".<br>**Notice!** In contrast to machine axes, only one axis can be traversed at a time with the traverse keys in the case of geometry axes.<br>Alarm 20062 is triggered if an attempt is made to traverse more than one axis with the traverse keys. |
| Signal state 0 or signal transition 1 ——> 0 | See cases 1 to 4 above |
| Signal irrelevant for ...... | Operating modes AUTOMATIC and MDA |
| Special cases, errors, ...... | The geometry axis cannot be traversed in JOG mode:<br>– If it is already being traversed via the axial PLC interface (as a machine axis).<br>– If another geometry axis is already being traversed with the traverse keys.<br>Alarm 20062 "Axis already active" is output. |
| Related to .... | IS "Traverse keys plus and minus" for machine axes (DB31, ... DBX8.7 or DBX8.6)<br>IS "Traverse key disable for geometry axes" (DB21, ... DBX12.4 ff) |

| | |
|---|---|
| **DB21, ...**<br>**DBX13, 17, 21**<br>**Bits 0–5**<br>Data Block | **Machine function for geometry axis (1,2,3)**<br>**INC1, INC10, INC100, INC1000, INC10000, INCvar**<br><br>Signal(s) to channel (PLC –> NCK) |
| Edge evaluation: no | Signal(s) updated: Cyclic / Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | This interface signal defines how many increments the geometry axis traverses when the traverse key is pressed or the handwheel is turned one detent position. JOG mode must be active for this (exception: with DRF).<br>The increment size is assigned to the interface signals as follows:<br>– for INC1 up to INC10000: with general machine data JOG_INCR_SIZE_TAB.<br>– for INCvar: with general setting data JOG_VAR_INCR_SIZE<br>As soon as the selected machine function becomes active, this is signaled to the PLC interface (interface signal “Active machine function INC1; ...” ).<br>If several machine function signals (INC1, INC... or “Continuous jogging”) are selected at the interface simultaneously, no machine function is activated by the control. |
| Signal state 0 or signal transition 1 ––> 0 | The machine function in question is not selected.<br>If an axis is currently traversing an increment, this movement is also aborted if this machine function is deselected or switched over. |
| Related to .... | IS “Active machine function INC1,...” for geometry axes (DB21, ... DBB41 ff)<br>IS “Machine function continuous” for geometry axes (DB21, ... DBX13.6 ff) |

| | |
|---|---|
| **DB21, ...**<br>**DBX13.6; 17.6; 21.6**<br>Data Block | **Machine function continuous for geometry axis**<br>**(1,2,3)**<br>Signal(s) to channel (PLC –> NCK) |
| Edge evaluation: no | Signal(s) updated: Cyclic / Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The machine function “Continuous jogging” is selected. The associated geometry axis can be traversed with the traverse keys plus and minus in JOG mode. |
| Signal state 0 or signal transition 1 ––> 0 | Machine function “Continuous jogging” is not selected. |
| Related to .... | IS “Active machine function INC 1,..., continuous” (DB21, ... DBB41 ff)<br>IS “Machine function INC1,...,INC10000” (DB21, ... DBB13 ff) |

## 5.2.3 Overview of signals from channel to PLC

| DB 21–28 | Signals from channel | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DBB | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 24 (MMC → PLC) | | Dry run feedrate selected | M01 selected | Single block selected | DRF selected | | | |
| 33 (MMC → PLC) | | | | | Handwheel override active | | | |
| 37 | Stop at the end of block with SBL suppressed | Read–in enable is ignored | CLC stopped upper limit /TE1/ | CLC stopped lower limit /TE1/ | CLC active /TE1/ | Contour handwheel active | | |
| | | | | | | Handwheel 1 | Handwheel 2 | Handwheel 3 |
| | Geometry axis 1 | | | | | | | |
| 40 | Motion command | | Traversing request | | | Handwheel active | | |
| | plus | minus | plus | minus | | 3 | 2 | 1 |
| 41 | Geometry axis 1 | | | | | | | |
| | Active machine function | | | | | | | |
| | | Continuous | Variable INC | 10000 INC | 1000 INC | 100 INC | 10 INC | 1 INC |
| | Geometry axis 2 | | | | | | | |
| 46 | Motion command | | Traversing request | | | Handwheel active | | |
| | plus | minus | plus | minus | | 3 | 2 | 1 |
| 47 | Geometry axis 2 | | | | | | | |
| | Active machine function | | | | | | | |
| | | Continuous | Variable INC | 10000 INC | 1000 INC | 100 INC | 10 INC | 1 INC |
| | Geometry axis 3 | | | | | | | |
| 52 | Motion command | | Traversing request | | | Handwheel active | | |
| | plus | minus | plus | minus | | 3 | 2 | 1 |
| 53 | Geometry axis 3 | | | | | | | |
| | Active machine function | | | | | | | |
| | | Continuous | Variable INC | 10000 INC | 1000 INC | 100 INC | 10 INC | 1 INC |

## 5.2.4 Description of signals from channel to PLC

| | | |
|---|---|---|
| **DB21, ...<br>DBX24.3**<br>Data Block | **DRF selected**<br>Signal(s) from channel (MMC –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The operator has selected DRF on the operator panel front. The PLC program (basic PLC program or user program) transfers this MMC interface signal as IS “Activate DRF” after logical combination.<br>As soon as DRF is active, the DRF offset can be changed in AUTOMATIC or MDA mode using the handwheel assigned to the axis. | |
| Signal state 0 or signal transition 1 ––> 0 | The operator has not selected DRF on the operator panel front. | |
| Signal irrelevant for ...... | JOG mode | |
| Related to .... | IS: “Activate DRF” (DB21, ... DBX0.3) | |

| | | |
|---|---|---|
| **DB21, ...<br>DBX33.3**<br>Data Block | **Handwheel override active**<br>Signal(s) from channel (NCK –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The function “Handwheel override in AUTOMATIC mode” is active for the programmed path axes.<br>Handwheel pulses of the 1st geometry axis function as a velocity override over the programmed path feedrate. | |
| Signal state 0 or signal transition 1 ––> 0 | The function “Handwheel override in AUTOMATIC mode” is not active for the programmed path axes.<br>An active handwheel override is deactivated again if<br>the path axes have reached the target position<br>the distance-to-go is deleted by the channel-specific IS “Delete distance-to-go” (DB21, ... DBX6.2).<br>a RESET is performed | |

| | | |
|---|---|---|
| **DB21, ...<br>DBX37<br>Bits 0–2**<br>Data Block | **Contour handwheel active (1 to 3)**<br>Signal(s) from channel (NCK –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 4.3 |
| Signal state 1 or signal transition 0 ––> 1 | These PLC interface signals report whether this geometry axis is assigned to contour handwheel 1, 2 or 3 or to no contour handwheel.<br>Only one contour handwheel can be assigned to an axis at any one time.<br>If several interface signals “Contour handwheel active” are set, priority<br>’Contour handwheel 1’ before ’Contour handwheel 2’ before ’Contour handwheel 3’ applies.<br>If the assignment is active, the geometry axis can be traversed in JOG mode with the contour handwheel or a DRF offset can be generated in AUTOMATIC or MDA modes. | |
| Signal state 0 or signal transition 1 ––> 0 | Neither contour handwheel 1, 2 nor 3 is assigned to this geometry axis. | |
| Related to .... | | |

| DB21, ...<br>**DBX40, 46, 52**<br>**Bits 0-2**<br>Data Block | **Handwheel active (1 to 3) for geometry axis**<br><br>Signal(s) from channel (NCK –> PLC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | These PLC interface signals report whether this geometry axis is assigned to handwheel 1, 2 or 3 or to no handwheel.<br>Only one handwheel can be assigned to an axis at any one time.<br>If several "Activate handwheel" interface signals are set, priority "Handwheel 1" before "Handwheel 2" before "Handwheel 3" applies.<br>If the assignment is active, the geometry axis can be traversed in JOG mode with the handwheel or a DRF offset can be generated in AUTOMATIC or MDA modes. | |
| Signal state 0 or signal transition 1 ——> 0 | Neither handwheel 1, 2 nor 3 is assigned to this geometry axis. | |
| Related to .... | IS "Activate handwheel" (DB21, ... DBX12.0 to DBX12.2 ff) | |

| **DB21, ...**<br>**DBX40, 46, 52**<br>**Bit 5, Bit 4**<br>Data Block | **Plus and minus traversing request (for geometry axis)**<br><br>Signal(s) from axis/spindle (NCK –> PLC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 7.2 |
| The signal is the same as the travel command signal in the earlier version. | | |
| Signal state 1 or signal transition 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:<br>– JOG mode: with the plus or minus traverse key<br>– REF mode: with the traverse key that takes the axis to the reference point<br>– AUTO/MDA mode: a program block containing a coordinate value for the axis in question is executed. | |
| Signal state 0 or signal transition 1 ——> 0 | A travel command in the relevant axis direction has not been given or a traverse movement has been completed.<br>JOG mode:<br>– The travel command is reset depending on the current setting "jog or continuous mode" (see interface signal "Traverse keys plus and minus"). For JOG with handwheel.<br>REF mode:<br>– When the reference point is reached<br>AUTO/MDA mode:<br>– The program block has been executed (and the next block does not contain any coordinate values for the axis in question)<br>– Abort with "RESET", etc.<br>– IS "Axis disable" is active | |
| Related to .... | IS "Travel command plus" and "Travel command minus"<br>(DB21, ... DBX40.7 or DBX40.6)<br>(DB21, ... DBX46.7 or DBX46.6)<br>(DB21, ... DBX52.7 or DBX52.6) | |

| | | |
|---|---|---|
| **DB21, ...**<br>**DBX40, 46, 52**<br>**Bits 7, 6**<br>Data Block | **Plus and minus travel commands (for geometry axis)**<br><br>Signal(s) from channel (NCK –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| **The signal operates as described if bit 0 in MD 17900: VDI_FUNCTION_MASK 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 signal traversing request plus/minus DB21, ... DBX 40, 46, 52 bits 5, 4, which is always output, has the same effect as signal travel command plus/ mins when MD 17900 bit 0 = 0.** | | |
| Signal state 1 or signal transition 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:<br>– JOG mode: with the plus or minus traverse key<br>– REF mode: with the traverse key that takes the axis to the reference point<br>– AUTO/MDA mode: the program block containing a coordinate value for the axis in question is executed. | |
| Signal state 0 or signal transition 1 ——> 0 | A travel command in the relevant axis direction has not been given or a traverse movement has been completed.<br>JOG mode:<br>– The travel command is reset depending on the current setting “jog or continuous mode” (see interface signal “Traverse keys plus and minus”).<br>– While traversing with the handwheel.<br>REF mode:<br>When the reference point is reached<br>AUTO/MDA mode:<br>– The program block has been executed (and the next block does not contain any coordinate values for the axis in question)<br>– Abort with “RESET”, etc.<br>– IS “Axis disable” is active | |
| Application example(s) | To release clamping of axes with clamping (e.g. on a rotary table).<br>**Note:** If the clamping is not released until the travel command is given, these axes cannot be operated under continuous path control! | |
| Related to .... | IS “Traverse key plus” and “Traverse key minus” for geometry axis (DB21, ... DBX12.7 or DBX12.6 ff)<br>IS “Traversing request plus/minus” (DB21, ... DBX 40, 46, 52 Bit 5, 4) | |

| | | |
|---|---|---|
| **DB21, ...**<br>**DBX41, 47, 53**<br>**Bits 0 –6**<br>Data Block | **Active machine function for geometry axis (1, 2, 3)**<br>**INC1, ..., continuous jogging**<br><br>Signal(s) from channel (NCK –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | The PLC interface receives a signal stating which JOG mode machine function is active for the geometry axes.<br>The reaction to actuation of the traverse key or rotation of the handwheel varies depending on which machine function is active (see Section 2.2 and 2.3). | |
| Signal state 0 or signal transition 1 ——> 0 | The machine function in question is not active. | |
| Related to .... | IS “Machine function INC1,...,continuous” for geometry axes (DB21, ... DBB13 ff) | |

<table>
<tr><td>DB21, ...<br>DBX332, 336, 340<br>Bit 5, Bit 4<br><br>Data Block</td><td colspan="2">Plus and minus traversing request (for orientation axis)<br><br><br>Signal(s) from axis/spindle (NCK –> PLC)</td></tr>
<tr><td>Edge evaluation: no</td><td>Signal(s) updated: Cyclic</td><td>Signal(s) valid from SW: 7.2</td></tr>
<tr><td colspan="3">The signal is the same as the travel command signal in the earlier version.</td></tr>
<tr><td>Signal state 1 or signal transition 0 ——> 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:<br>– JOG mode: with the plus or minus traverse key<br>– REF mode: with the traverse key that takes the axis to the reference point<br>– AUTO/MDA mode: a program block containing a coordinate value for the axis in question is executed.</td></tr>
<tr><td>Signal state 0 or signal transition 1 ——> 0</td><td colspan="2">A travel command in the relevant axis direction has not been given or a traverse movement has been completed.<br>JOG mode:<br>– The travel command is reset depending on the current setting “jog or continuous mode” (see interface signal “Traverse keys plus and minus”). For JOG with handwheel.<br>...REF mode:<br>– When the reference point is reached<br>...AUTO/MDA mode:<br>– The program block has been executed (and the next block does not contain any coordinate values for the axis in question)<br>– Abort with “RESET”, etc.<br>– IS “Axis disable” is active</td></tr>
<tr><td>Related to ....</td><td colspan="2">IS “Travel command plus” and “Travel command minus”<br>(DB31, ... DBX332.7 or DBX332.6)<br>(DB31, ... DBX336.7 or DBX336.6)<br>(DB31, ... DBX340.7 or DBX340.6)</td></tr>
</table>

<table>
<tr><td>DB21, ...<br>DBX332, 336, 340<br>Bit 7, Bit 6<br>Data Block</td><td colspan="2">Plus and minus travel command (for orientation axis)<br><br>Signal(s) from channel (NCK –> PLC)</td></tr>
<tr><td>Edge evaluation: no</td><td>Signal(s) updated: Cyclic</td><td>Signal(s) valid from SW: 1.1</td></tr>
<tr><td colspan="3">The signal operates as described if bit 0 in MD 17900: VDI_FUNCTION_MASK 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 signal traversing request plus/minus DB21, ... DBX 332, 336, 340 bits 5, 4, which is always output, has the same effect as signal travel command plus/mins when MD 17900 bit 0 = 0.</td></tr>
<tr><td>Signal state 1 or signal transition 0 ——> 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:<br>– JOG mode: with the plus or minus traverse key<br>– REF mode: with the traverse key that takes the axis to the reference point<br>– AUTO/MDA mode: the program block containing a coordinate value for the axis in question is executed.</td></tr>
</table>

| | |
|---|---|
| **DB21, ...**<br>**DBX332, 336, 340**<br>**Bit 7, Bit 6**<br>Data Block | **Plus and minus travel command (for orientation axis)**<br><br>Signal(s) from channel (NCK –> PLC) |
| Signal state 0 or signal transition 1 ––> 0 | A travel command in the relevant axis direction has not been given or a traverse movement has been completed.<br>JOG mode:<br>– The travel command is reset depending on the current setting "jog or continuous mode" (see interface signal "Traverse keys plus and minus").<br>– While traversing with the handwheel.<br>REF mode:<br>When the reference point is reached<br>AUTO/MDA mode:<br>– The program block has been executed (and the next block does not contain any coordinate values for the axis in question)<br>– Abort with "RESET", etc.<br>– IS "Axis disable" is active |
| Application example(s) | To release clamping of axes with clamping (e.g. on a rotary table).<br>**Note:** If the clamping is not released until the travel command is given, these axes cannot be operated under continuous path control! |
| Related to .... | IS "Traverse key plus" and "Traverse key minus" for geometry axis (DB21, ... DBX12.7 or DBX12.6 ff)<br>IS "Traversing request plus/minus" (DB21, ... DBX 332, 336, 340 Bit 5, 4) |

| | | |
|---|---|---|
| **DB21, ...**<br>**DBX41, 47, 53**<br>**Bits 0 –6**<br>Data Block | **Active machine function for geometry axis (1, 2, 3)**<br>**INC1, ..., continuous jogging**<br><br>Signal(s) from channel (NCK –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The PLC interface receives a signal stating which JOG mode machine function is active for the geometry axes.<br>The reaction to actuation of the traverse key or rotation of the handwheel varies depending on which machine function is active (see Section 2.2 and 2.3). | |
| Signal state 0 or signal transition 1 ––> 0 | The machine function in question is not active. | |
| Related to .... | IS "Machine function INC1,...,continuous" for geometry axes (DB21, ... DBB13 ff) | |

### 5.2.5 Description of signals for contour handwheel

**Overview of interface signals for contour handwheel**

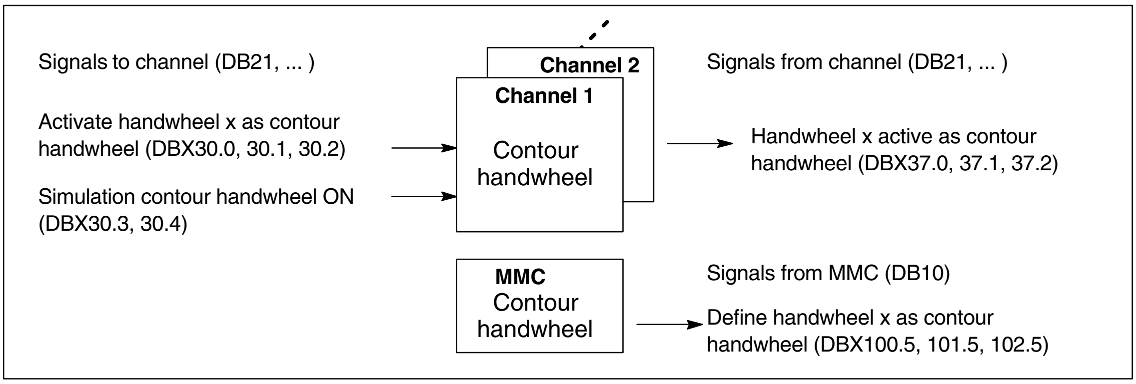



Fig. 5-1 Overview of interface signals for contour handwheel

| **DB 21, 22, ...**<br>**DBX30.0**<br>**DBX30.1**<br>**DBX30.2**<br>**Data Block** | **Activate handwheel 1 as contour handwheel**<br>**Activate handwheel 2 as contour handwheel**<br>**Activate handwheel 3 as contour handwheel**<br>**Signal(s) to channel (PLC → NCK)** | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW:<br>840D SW4.1, 810D SW2.1 |
| Description | These signals allow one of the three handwheels to be selected/deselected as the contour handwheel.<br><br>Signal = 1 Handwheel x is selected as the contour handwheel<br>Signal = 0 Handwheel x is deselected as the contour handwheel<br><br>The contour handwheel can be selected/deselected in the middle of a block.<br>When the handwheel is activated, the axis movement is first decelerated and then traversed as determined by the handwheel.<br>When the handwheel is deactivated, the movement is first decelerated before execution of the NC program continues. 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 maintained after NC RESET. | |
| Related to .... | IS “Handwheel x active as contour handwheel” (DB21, 22, ... , DBX37.0, 37.1, 37.2) | |

| | | |
|---|---|---|
| **DB 21, 22, ...**<br>**DBX30.3**<br>**DBX30.4**<br>Data Block | **Simulation contour handwheel on**<br>**Negative direction simulation contour handwheel**<br>Signal(s) to channel (PLC —> NCK) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW:<br>840D SW4.1, 810D SW2.1 |
| Description | To activate/deactivate simulation of the contour handwheel and to set the traversing direction, these signals must be set as follows:<br><br>Bit 3 Bit 4 Meaning<br>0 0 Simulation OFF<br>0 1 Simulation OFF<br>1 0 Simulation ON, direction as programmed<br>1 1 Simulation ON, opposite direction to programmed direction<br><br>During simulation, the feedrate is not determined by the contour handwheel, but the axis is traversed along the contour at the programmed feedrate.<br>If the function is deselected, the current axis movement is decelerated along a braking ramp.<br>When the traversing direction is reversed, the current axis movement is decelerated along a braking ramp and the axis then traversed in the opposite direction. | |
| Special cases, errors, ...... | Simulation is only effective in AUTOMATIC mode and can only be activated when the contour handwheel is activated. | |

| | | |
|---|---|---|
| **DB 21, 22, ...**<br>**DBX37.0**<br>**DBX37.1**<br>**DBX37.2**<br>Data Block | **Handwheel 1 active as contour handwheel**<br>**Handwheel 2 active as contour handwheel**<br>**Handwheel 3 active as contour handwheel**<br>Signal(s) from channel (NCK —> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW:<br>840D SW4.1, 810D SW2.1 |
| Description | These signals indicate which handwheel is selected as the contour handwheel.<br><br>Signal = 1 Handwheel x is selected as the contour handwheel<br>Signal = 0 Handwheel x is deselected as the contour handwheel | |
| Special cases, errors, ...... | The signal is maintained after NC RESET. | |
| Related to .... | IS “Activate handwheel x as contour handwheel” (DB21, 22, ... DBX30.0, 30.1, 30.2) | |

| | | |
|---|---|---|
| **DB 10**<br>**DBX100.5**<br>**DBX101.5**<br>**DBX102.5**<br>Data Block | **Define handwheel 1 as contour handwheel**<br>**Define handwheel 2 as contour handwheel**<br>**Define handwheel 3 as contour handwheel**<br>Signal(s) from MMC (MMC –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW:<br>840D SW4.1, 810D SW2.1 |
| Description | These signals indicate which handwheel is defined as the contour handwheel via the MMC.<br><br>Signal = 1 Handwheel x is defined as the contour handwheel via the MMC<br>Signal = 0 Handwheel x is not defined as the contour handwheel<br><br>To ensure that the handwheel defined via the MMC can operate as the contour handwheel, the appropriate signal must be gated with IS “Activate handwheel x as contour handwheel” (DB21, 22, ... ,DBX30.0, 30.1, 30.2). | |
| 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. | |
| Related to .... | IS “Activate handwheel x as contour handwheel” (DB21, 22, ..., DBX30.0, 30.1, 30.2)<br>FB1 parameter HWheelMMC | |

# 5.3 Axis/spindle-specific signals

## 5.3.1 Overview of signals to axis/spindle

| DB 31, ... | Signals to axis/spindle | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DBB | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 4 | Traversing keys plus | Traversing keys minus | Rapid traverse override | Traversing key lock | Feed hold Spindle hold | Activate handwheel 3 | Activate handwheel 2 | Activate handwheel 1 |
| 5 | Machine function | | | | | | | |
| | | Continuous | Variable INC | 10000 INC | 1000 INC | 100 INC | 10 INC | 1 INC |

## 5.3.2 Description of signals to axis/spindle

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX4.0; 4.1; 4.2**<br>Data Block | **Activate handwheel (1 to 3)**<br>Signal(s) to axis/spindle (PLC –> NCK) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | This PLC interface signal defines whether this machine axis is assigned to handwheel 1, 2, 3 or no handwheel.<br>Only one handwheel can be assigned to an axis at any one time.<br>If several “Activate handwheel” interface signals are set, priority “Handwheel 1” before “Handwheel 2” before “Handwheel 3” applies.<br>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 signal transition 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. | |
| Related to .... | IS “Handwheel active” (DB31, ... DBX64.0 to DBX64.2) | |

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX4.4**<br>Data Block | **Traverse key disable**<br>Signal(s) to axis/spindle (PLC –> NCK) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 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.<br>If the traverse key disable is activated during a traverse movement, the machine axis is stopped. | | |
| Signal state 0 or signal transition 1 ––> 0 | Traverse keys plus and minus 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. | | |
| Related to .... | IS “Traverse key plus” and “Traverse key minus” (DB31, ... DBX4.7 or DBX4.6) | | |

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX4.5**<br>Data Block | **Rapid traverse override**<br>Signal(s) to axis/spindle (PLC –> NCK) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | If interface signal “Rapid traverse override” is set together with “Traverse key plus” and “Traverse key minus”, the machine axis in question traverses at rapid traverse.<br>The rapid traverse feedrate is defined in machine data JOG_VELO_RAPID.<br>Rapid traverse override is active in the following JOG modes:<br>– Continuous jogging<br>– Incremental jogging<br>If rapid traverse override is active, the velocity can be modified with the rapid traverse override switch. | | |
| Signal state 0 or signal transition 1 ––> 0 | The machine axis traverses at the defined JOG velocity (SD: JOG_SET_VELO or MD: JOG_VELO). | | |
| Signal irrelevant for ...... | – Operating modes AUTOMATIC and MDA<br>– Reference point approach (JOG mode) | | |
| Related to .... | IS “Traverse key plus” and “Traverse key minus” (DB31, ... DBX4.7 or DBX4.6)<br>IS “Axial feedrate/spindle speed override” (DB31, ... DBB0) | | |

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX4.7, 4.6**<br>Data Block | **Plus and minus traverse keys**<br>Signal(s) to axis/spindle (PLC –> NCK) | | |
| Edge evaluation: yes | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The selected machine axis can be traversed in both directions in JOG mode with the traverse keys plus and minus.<br>Depending on the active machine function and the setting “Jog or continuous mode” (SD: JOG_CONT_MODE_LEVELTRIGGRD for JOG continuous and MD: JOG_INC_MODE_LEVELTRIGGRD for JOG INCR), the signal transition will cause different reactions.<br>Case 1: **Continuous jogging with jog mode**<br>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).<br>Case 2: **Continuous jogging with continuous mode**<br>On the first signal edge change from 0→ 1 the machine axis starts to traverse in the relevant direction. This traversing movement still continues when the edge changes from 1 → 0. A new signal edge change from 0 → 1 (same traversing direction!) stops the traversing movement.<br>Case 3: **Incremental jogging with jog mode**<br>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.<br>Case 4: **Incremental jogging with continuous mode**<br>On the first signal edge change from 0 → 1 the machine axis starts to traverse the set increment. If the same traversing signal is applied and the edge changes from 0 → 1 again before the axis has traversed the increment, the traverse movement is aborted. The increment is not traversed to the end.<br>If both traverse signals (plus and minus) are set at the same time there is no movement or a current movement is aborted.<br>The effect of the traverse keys can be disabled for every machine axis individually with the PLC interface signal “Traverse key disable”. | | |
| Signal state 0 or signal transition 1 ––> 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).<br>Alarm 20062 is signaled. | | |
| Special cases, ...... | Indexing axes | | |
| Related to .... | IS “Traverse keys plus and minus for geometry axes” (DB21, ... DBX12.7 and DBX12.6 ff)<br>IS “Traverse key disable” (DB31, ... DBX4.4 ) | | |

| DB31, ... DBX5 Bits 0 –5 Data Block | **Machine function INC1, INC10, INC100, INC1000, INC10000, INCvar** Signal(s) to axis/spindle (PLC –> NCK) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | This interface signal defines how many increments the machine axis traverses when the traverse key is pressed or the handwheel is turned one detent position. JOG mode must be active for this (exception: with DRF).<br>The increment size is assigned to the interface signals as follows:<br>– INC1 to INC10000: with general machine data JOG_INCR_SIZE_TAB.<br>– INCvar: with general setting data JOG_VAR_INCR_SIZE<br>As soon as the selected machine function becomes active, this is signaled to the PLC interface (interface signal “Active machine function INC1; ...” ).<br>If several machine function signals (INC1, INC... or “Continuous jogging”) are selected at the interface simultaneously, no machine function is activated by the control. | |
| Signal state 0 or signal transition 1 ––> 0 | The machine function in question is not selected.<br>If an axis is currently traversing an increment, this movement is also aborted if this machine function is deselected or switched over. | |
| Related to .... | IS “Active machine function INC1,...” (DB31, ... DBB65)<br>IS “Machine function continuous” (DB31, ... DBX5.6). | |

| DB31, ... DBX5.6 Data Block | **Continuous machine function** Signal(s) to axis/spindle (PLC –> NCK) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The machine function “Continuous jogging” is selected. The associated machine axis can be traversed with the traverse keys plus and minus in JOG mode. | |
| Signal state 0 or signal transition 1 ––> 0 | Machine function “Continuous jogging” is not selected. | |
| Related to .... | IS “Active machine function INC 1,..., continuous” (DB31, ... DBB65)<br>IS “Machine function INC1,...,INC10000” (DB31, ... DBB5) | |

## 5.3.3 Overview of signals from axis/spindle

| DB 31, ... | Signals from axis/spindle | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DBB | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 64 | Motion command | | Traversing request | | | Handwheel active | | |
| | plus | minus | plus | minus | | 3 | 2 | 1 |
| 65 | Active machine function | | | | | | | |
| | | Continu-ous | Variable INC | 10000 INC | 1000 INC | 100 INC | 10 INC | 1 INC |

## 5.3.4 Description of signals from axis/spindle

| DB31, ... **DBX62.1** Data Block | **Handwheel override active** Signal(s) from axis/spindle (NCK –> PLC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The function "Handwheel override in AUTOMATIC mode" is active for the programmed positioning axis (FDA[AXi]). Handwheel pulses for this axis either act as a path setting (if FDA=0) or as a velocity override (if FDA>0) over the programmed axis feedrate.<br>The interface signal is also set if "Handwheel override in AUTOMATIC mode" is active for a concurrent positioning axis (with FC15). | |
| Signal state 0 or signal transition 1 ––> 0 | The function "Handwheel override in AUTOMATIC mode" is not active for the programmed positioning axis (or concurrent positioning axis).<br>An active handwheel override is not active if<br>• The positioning axis has reached the target position<br>• the distance-to-go is deleted by the axial IS "Delete distance-to-go" (DB31, ... DBX2.2).<br>• a RESET is performed | |

| **DB31, ... DBX64.0; 64.1; 64.2** Data Block | **Handwheel active (1 to 3)** Signal(s) from axis/spindle (NCK –> PLC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | These PLC interface signals provide feedback whether the machine axis is assigned to handwheel 1, 2, 3 or no handwheel.<br>Only one handwheel can be assigned to an axis at any one time.<br>If several "Activate handwheel" interface signals are set, priority "Handwheel 1" before "Handwheel 2" before "Handwheel 3" applies.<br>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 signal transition 1 ––> 0 | Neither handwheel 1, 2 nor 3 is assigned to this geometry axis. | |
| Related to .... | IS "Activate handwheel" (DB31, ... DBX4.0 to DBX4.2)<br>IS "Handwheel selected" (DB10, DBB100.6 ff) | |

| DB31, ...<br>**DBX64.5, 64.4**<br>Data Block | **Plus and minus traversing request**<br>Signal(s) from axis/spindle (NCK –> PLC) |
|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic — Signal(s) valid from SW: 7.2 |
| The signal is the same as the travel command signal in the earlier version. | |
| Signal state 1 or signal transition 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:<br>– JOG mode: with the plus or minus traverse key<br>– REF mode: with the traverse key that takes the axis to the reference point<br>– AUTO/MDA mode: the program block containing a coordinate value for the axis in question is executed. |
| Signal state 0 or signal transition 1 ––> 0 | A travel command in the relevant axis direction has not been given or a traverse movement has been completed.<br>JOG mode:<br>– The travel command is reset depending on the current setting “jog or continuous mode” (see interface signal “Traverse keys plus and minus”).<br>– While traversing with the handwheel.<br>– REF mode: When the reference point is reached<br>AUTO/MDA mode:<br>– The program block has been executed (and the next block does not contain any coordinate values for the axis in question)<br>– Abort with “RESET”, etc.<br>– IS “Axis disable” is active |
| Application example(s) | To release clamping of axes with clamping (e.g. on a rotary table).<br>**Note:** If the clamping is not released until the travel command is given, these axes cannot be operated under continuous path control! |
| Related to .... | IS “Traverse key plus” and “Traverse key minus” (DB31, ... DBX4.7 or DBX4.6)<br>IS “Travel command plus and minus” DB31, ... DBX64.7 or DBX64.6 |

| DB31, ...<br>**DBX64.7, 64.6**<br>Data Block | **Plus and minus traverse keys**<br>Signal(s) from axis/spindle (NCK –> PLC) |
|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic — Signal(s) valid from SW: 1.1 |
| **The signal operates as described if bit 0 in MD 17900: VDI_FUNCTION_MASK 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 signal traversing request plus/minus DB31, ... DBX 64, 46, 52 bits 5, 4, which is always output, has the same effect as signal travel command plus/ mins when MD 17900 bit 0 = 0.** | |
| Signal state 1 or signal transition 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:<br>– JOG mode: with the plus or minus traverse key<br>– REF mode: with the traverse key that takes the axis to the reference point<br>– AUTO/MDA mode: the program block containing a coordinate value for the axis in question is executed. |
| Signal state 0 or signal transition 1 ––> 0 | A travel command in the relevant axis direction has not been given or a traverse movement has been completed.<br>JOG mode:<br>– The travel command is reset depending on the current setting “jog or continuous mode” (see interface signal “Traverse keys plus and minus”).<br>– While traversing with the handwheel.<br>– REF mode: When the reference point is reached<br>AUTO/MDA mode:<br>– The program block has been executed (and the next block does not contain any coordinate values for the axis in question)<br>– Abort with “RESET”, etc.<br>– IS “Axis disable” is active |
| Application example(s) | To release clamping of axes with clamping (e.g. on a rotary table).<br>**Note:** If the clamping is not released until the travel command is given, these axes cannot be operated under continuous path control! |
| Related to .... | IS “Traverse key plus” and “Traverse key minus” (DB31, ... DBX4.7 or DBX4.6)<br>IS “Traversing request plus and minus” (DB31, ... DBX64.5 or DBX.4) |

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX65 Bits 0 –6**<br>Data Block | **Active machine function**<br>**INC1, ..., continuous jogging**<br>Signal(s) from axis/spindle (NCK –> PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | The PLC interface receives a signal stating which JOG mode machine function is active for the machine axes.<br>The result when the traverse key is pressed or the handwheel is turned depends on the active machine function (see Sections 2.2 and 2.3). | | |
| Signal state 0 or signal transition 1 ——> 0 | The machine function in question is not active. | | |
| Related to .... | IS “Machine function INC1,...,continuous” (DB31, ... DBB5) | | |

■

# Notes

# Example 6

None

# Data Fields, Lists 7

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **Signals to/from NC** | | | |
| 10 | 97, 98, 99 | Channel number for geometry axis handwheel 1, 2, 3 | |
| 10 | 100, 101, 102 | Axis number for handwheel 1, 2, 3, handwheel selected and machine axis | |
| **Mode groupspecific** | | | |
| 11, ... | 0.2 | JOG mode | K1 |
| 11, ... | 4.2 | Active JOG mode | K1 |
| **Channel-specific** | | | |
| 21, ... | 0.3 | Activate DRF | |
| 21, ... | 12.2, 12.1, 12.0<br>16.2, 16.1, 16.0<br>20.2, 20.1, 20.0 | Activate handwheel 1, 2, 3 | |
| 21, ... | 12.4, 16.4, 20.4 | Traversing key lock | |
| 21, ... | 12.5, 16.5, 20.5 | Rapid traverse override | |
| 21, ... | 12.7, 12.6<br>16.7, 16.6<br>20.7, 20.6 | Traverse keys plus and traverse keys minus | |
| 21, ... | 13, 17, 21 | Geometry axis machine function INC1 ... continuous | |
| 21, ... | 24.3 | DRF selected | |
| 21, ... | 40.2, 40.1, 40.0<br>46.2, 46.1, 46.0<br>52.2, 52.1, 52.0 | Handwheel active (3, 2, 1) | |
| 21, ... | 40.5, 40.4<br>46.5, 46.4<br>52.5, 52.4 | Traversing request geometry axis | |

| Channel-specific | | | |
|---|---|---|---|
| 21, ... | 40.7, 40.6<br>46.7, 46.6<br>52.7, 52.6 | Travel command plus and travel command minus | |
| 21, ... | 41, 47, 53 | Geometry axis active machine function INC1 ... continuous jogging | |
| 21, ... | 33.3 | Handwheel override active for path axes (SW2 and higher) | |
| 21, ... | 30.0<br>30.1<br>30.2 | Activate handwheel 1 as contour handwheel<br>Activate handwheel 2 as contour handwheel<br>Activate handwheel 3 as contour handwheel | |
| 21, ... | 30.3 | Simulation contour handwheel on | |
| 21, ... | 30.4 | Negative direction simulation contour handwheel | |
| 21, ... | 37.0<br>37.1<br>37.2 | Handwheel 1 active as contour handwheel<br>Handwheel 2 active as contour handwheel<br>Handwheel 3 active as contour handwheel | |
| 21, ... | 100.5<br>101.5<br>102.5 | Handwheel 1 active as contour handwheel<br>Handwheel 2 active as contour handwheel<br>Handwheel 3 active as contour handwheel | |
| 21, ... | 332.5, 332.4<br>336.5, 336.4<br>340.5, 340.4 | Traversing request orientation axis | |
| **Axis/spindle-specific** | | | |
| 31, ... | 0 | Feedrate/spindle override | V1 |
| 31, ... | 1.7 | Override active | V1 |
| 31, ... | 2.2 | Axial delete distance-to-go | |
| 31, ... | 4.2, 4.1, 4.0 | Activate handwheel 1, 2, 3 | |
| 31, ... | 4.4 | Traversing key lock | |
| 31, ... | 4.5 | Rapid traverse override | |
| 31, ... | 4.7, 4.6 | Traverse keys plus and traverse keys minus | |
| 31, ... | 5.6 | Continuous machine function | |
| 31, ... | 5.6, 5.5, 5.4, 5.3,<br>5.2, 5.1, 5.0 | Machine function<br>continuous, Var. INC, 10000 INC,<br>1000 INC, 100 INC, 10 INC, 1 INC | |
| 31, ... | 60.7, 60.6 | Position reached with exact stop coarse/fine | B1 |
| 31, ... | 64.2, 64,1, 64.0 | Handwheel active (3, 2, 1) | |
| 31, ... | 64.5, 64.4 | Plus and minus traversing request axis/spindle | |
| 31, ... | 64.7, 64.6 | Travel command plus and travel command minus | |
| 31, ... | 65 | Active machine function INC1 ... continuous jogging | |
| 31, ... | 62.1 | Handwheel override active, for positioning axes and concurrent positioning axes (SW2 and higher) | |

# 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ...** | | | |
| 10000 | AXCONF_MACHAX_NAME_TAB[n] | Machine axis name [n = axis number] | K2 |
| 11300 | JOG_INC_MODE_LEVELTRIGGRD | INC and REF in jog mode | |
| 11310 | HANDWH_REVERSE | Defines movement in the opposite direction | |
| 11320 | HANDWH_IMP_PER_LATCH[n] | Handwheel pulses per detent position [n=handwheel number: 0 – 2] | |
| 11330 | JOG_INCR_SIZE_TAB[n] | Increment size INC/handwheel (n = increment index: 0 – 4) | |
| 11340 | ENC_HANDWHEEL_SEGMENT_NR | Third handwheel: Bus segment | FBMA |
| 11342 | ENC_HANDWHEEL_MODULE_NR | Third handwheel: Drive no./measuring circuit no. | FBMA |
| 11344 | ENC_HANDWHEEL_INPUT_NR | Third handwheel: input on module/measuring circuit card | |
| 11346 | HANDWH_TRUE_DISTANCE | Handwheel path or velocity values | FBMA |
| 17900 | VDI_FUNCTION_MASK | Function mask for VDI signals | |
| **Channelspecific ($MC_ ... )** | | | |
| 20060 | AXCONF_GEOAX_NAME_TAB[n] | Geometry axis in channel [n = geometry axis number] | K2 |
| 20100 | DIAMETER_AX_DEF | Geometry axes with transverse axis functions | P1 |
| 20620 | HANDWH_GEOAX_MAX_INCR_SIZE | Delimitation of the geometry axis | |
| 20622 | HANDWH_GEOAX_MAX_INCR_VSIZE | Path velocity override | |
| 20624 | HANDWH_CHAN_STOP_COND | Response to channel-specific VDI interface signals bits 0...7 | |
| **Axis/channelspecific ($MA_ ... )** | | | |
| 30450 | IS_CONCURRENT_POS_AX | Default setting at reset: Neutral axis or channel axis | P2 |
| 31090 | JOG_INCR_WEIGHT | Evaluation of an increment for INC/handwheel | |
| 32000 | MAX_AX_VELO | Maximum axis velocity | G2 |
| 32010 | JOG_VELO_RAPID | Conventional rapid traverse | |
| 32020 | JOG_VELO | Conventional axis velocity | |
| 32040 | JOG_REV_VELO_RAPID | Revolutions feedrate in JOG mode with rapid traverse override | |
| 32050 | JOG_REV_VELO | Revolutional feedrate in JOG mode | |
| 32060 | POS_AX_VELO | Initial setting for positioning-axis velocity | P2 |
| 32080 | HANDWH_MAX_INCR_SIZE | Limitation of the size of the selected increment | |
| 32082 | HANDWH_MAX_INCR_VELO_SIZE | Limitation of selected increment for velocity override | |
| 32084 | HANDWH_STOP_COND | Effect of axis-specific VDI interface signal bits 0...5 on the handwheel | |
| 32090 | HANDWH_VELO_OVERLAY_FACTOR | Ratio JOG velocity to handwheel velocity (with DRF) | |
| 35130 | GEAR_STEP_MAX_VELO_LIMIT[n] | Maximum velocity for gear stage | S1 |

## 7.3 Setting data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **General ($SN_ ...)** | | | |
| 41010 | JOG_VAR_INCR_SIZE | Size of variable increment for INC/handwheel | |
| 41050 | JOG_CONT_MODE_LEVELTRIGGRD | JOG continuous 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 speed for rotary axes | |
| 41200 | JOG_SPIND_SET_VELO | JOG velocity for the spindle | |

## 7.4 Alarms

For detailed descriptions of the alarms, please refer to
**References:** /DA/, “Diagnostics Guide”
and the online help of MMC 101/102/103 systems.

■

# SINUMERIK 840D/810D Description of Functions Extended Functions (FB2)

# Compensations (K3)

■

# Brief Description 1

**Reason**

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.

Some of these deviations can usually be measured during installation and then compensated for during operation on the basis of values read by the positional actual-value encoder and other sensory devices.

**Compensation**

CNCs provide functions for compensation of the essential causes of error to meet the increasing demand for precision in machine tools.

For SINUMERIK 840D the following axis-specific compensations can be activated:

- Temperature compensation
- Backlash compensation
- Interpolatory compensation
  - LEC
    (leadscrew error and measuring system error compensation)
  - Beam sag compensation
    (compensation of beam sag and angular errors)
- Dynamic feedforward control (following error compensation)
- Friction compensation (or quadrant error compensation)
  - Conventional friction compensation
  - Quadrant error compensation with neural networks
    (SINUMERIK 840C only)
- Electronic counterweight for drives on SIMODRIVE 611D

These compensation functions can be set for each machine individually with axis-specific machine data.

**Interpolatory compensation**

The “Interpolatory compensation” function allows position-related dimensional deviations (for example, by leadscrew errors, measuring system errors or sag) to be corrected.

The compensation values are measured during installation and stored in a table as a position-related value. During operation the axis is compensated between interpolation points during linear interpolation.

**Friction compensation**

The “friction compensation” function is particularly effective in achieving a significant improvement in contour accuracy in circular contour machining operations. If the direction of rotation of an axis changes, contour errors occur when the velocity equals zero (quadrant transition point) because of the changing friction conditions. “Friction compensation” (also called “Quadrant error compensation”) compensates for this error reliably the first time the contour is machined.

A **neural network** integrated in the SINUMERIK 840D adapts the optimum parameters in a self-learning process to compensate for friction, backlash or torsion. The system allows for simple, automatic re-optimization at any time.

The friction compensation system is installed most simply with a circularity test. The circular contour is followed and the actual position deviations from the programmed radius (most especially at the quadrant transition points) are measured and then displayed graphically. The circularity test is an “installation tool” function.

**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 “Service axes” display in the “Diagnosis” operating area.

■

# Detailed Description 2

## 2.1 Temperature compensation

### 2.1.1 General

**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. The degree of expansion depends on the temperature and the thermal conductivity of the machine parts.

**Effects**

Owing to the thermal expansion of the machinery, the actual positions of the axes change depending on temperature. Since this phenomenon impairs the accuracy of the machined workpieces, it is possible to compensate such temperature-related changes in actual value position (so-called temperature compensation).

**Sensors**

Apart from the position actual values supplied by existing encoders, temperature compensation functions generally require a number of additional temperature sensors to acquire a temperature profile.
As temperature-related changes take a relatively long time to have an effect, acquisition and preprocessing of the temperature profile can be executed by the PLC in one-minute cycles.

**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**

The error curve characteristic shown in the figure below is frequently obtained.

If a position reference point $P_0$ is chosen for the axis, 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 (see figure below).

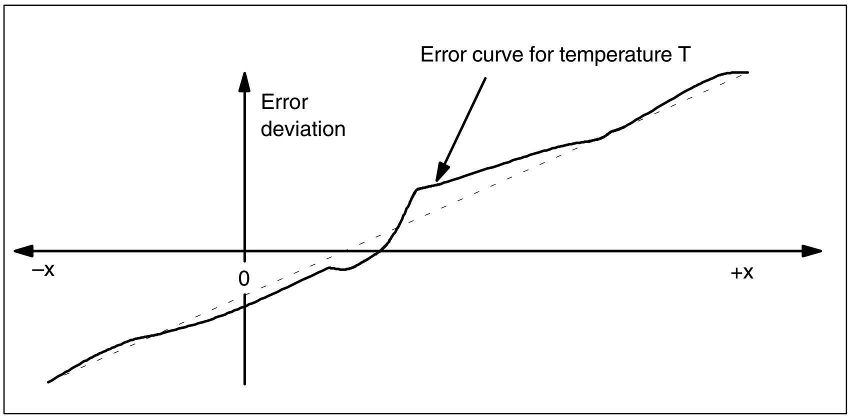



Fig. 2-1 Example of an error curve for heat expansion

**Compensation equation**

The compensation value $\Delta K_x$ is calculated on the basis of current actual position $P_x$ of this axis and temperature T according to the following equation:

$$\Delta K_x = K_0 (T) + \tan\beta (T) * (P_x - P_0)$$

Key to letters (see figure below):

$\Delta K_x$ 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
$\tan\beta$ Coefficient of the position-dependent temperature compensation (corresponds to the gradient of the approximated error line)

The compensation values are acquired in interpolation cycles. If the compensation value $\Delta K_x$ is positive, the axis moves in the negative direction.

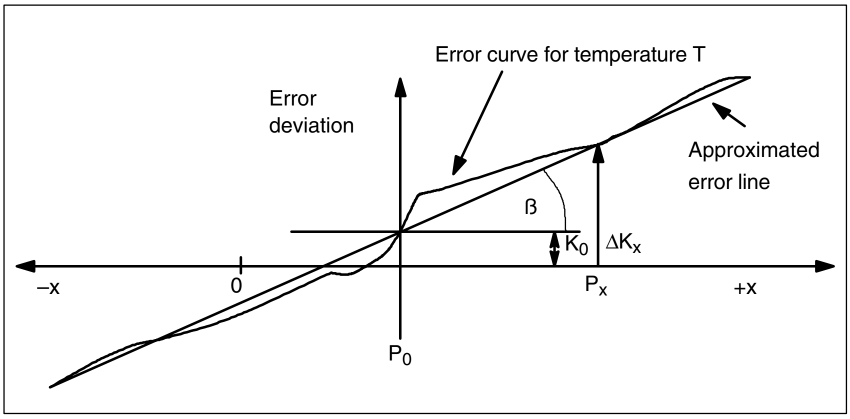



Fig. 2-2 Approximated error line for temperature compensation

## 2.1.2 Temperature compensation parameters

**Temperature-dependent parameters**

Error curves for different temperatures can be defined for each axis, as illustrated in the figure above. For each error curve the following parameters must be determined and then entered in the setting data:

- Position-dependent temperature-compensation value $K_0$
  SD 43900: TEMP_COMP_ABS_VALUE
- Reference position $P_0$ for position-dependent temperature compensation SD 43920: TEMP_COMP_REF_POSITION
- Slope tanβ for position-dependent temperature compensation SD 43910: TEMP_COMP_SLOPE

**Activate temperature compensation**

Temperature compensation can be activated for every axis by means of axial MD 32750: TEMP_COMP_TYPE. The type of temperature compensation to be applied can also be selected and this can be activated for several compensation types simultaneously.

Table 2-1 MD 32750: TEMP_COMP_TYPE

| MD 32750: TEMP_COMP_TYPE | Meaning | Associated parameters |
|---|---|---|
| Value = 0 | No temperature compensation active | ________________________ |
| Bit 0 = 1 | Position-independent temperature compensation active | SD 43900: TEMP_COMP_ABS_VALUE |
| Bit 1 = 1 | Position-dependent temperature compensation active | SD 43920: TEMP_COMP_REF_POSITION<br>SD 43910: TEMP_COMP_SLOPE. |
| Bit 2 = 1 | Temperature compensation active in tool direction | MD 20390: TOOL_TEMP_COMP_ON<br>For further details see:<br>**References:**<br>/FB/, W1 Section 2.8 |

**Activation**

The following conditions must be fulfilled before temperature compensation can be applied:

1. The option must be enabled.
2. The compensation type must be selected (MD 32750: TEMP_COMP_TYPE).
3. The parameters for the compensation type are defined.
4. The axis must be referenced (IS “Referenced/synchronized 1 or 2” DB31 to 48, DBX60.4 or 60.5 = ‘1’).

As soon as these conditions are fulfilled, the temperature compensation value for the current position actual value is added to the setpoint in all modes and the machine axis is traversed.

If the reference position is subsequently lost again, e.g. because the encoder frequency has been exceeded (IS “Referenced/Synchronized 1 or 2” = 0), then the compensation processing routine is aborted.

**Temperature fluctuations and modify parameters**

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 be compensated for effectively.

When temperature T changes, the parameters which are temperature-dependent, i.e. ($K_0$, $\tan\beta$ and $P_0$) also change and can thus always be overwritten by the PLC or by means of a synchronous action.

It is thus possible for the machine-tool manufacturer to represent 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 with variable Services (FB2 (GET) “Read data” and FB3 (PUT) “Write data”).

For more information on handling and parameterization of FB2 and FB3 see:
**References:** /FB/, P3 “Basic PLC Program”.

**Monitoring functions**

Axial MD 32760: COMP_ADD_VELO_FACTOR (velocity violation due to compensation) can be set to limit the maximum compensation value that can be added to the specified velocity value in each IPO clock cycle.

This machine data limits the maximum gradient of the error curve. If the maximum gradient is exceeded, the compensation value is limited in the control.

**Smooth compensation value**

To prevent overloading of the machine or tripping of monitoring functions in response to step changes in the above parameter settings, the compensation values are distributed among several IPO clock cycles by an internal control function as soon as they exceed the maximum compensation value specified for each cycle (MD 32760: COMP_ADD_VELO_FACTOR).

**Position display**

The normal actual value and setpoint position displays ignore the compensation values and show the position values of an ideal machine.

**Display of compensation values**

The total compensation value calculated from the temperature and sag compensation functions belonging to the current actual position is output in the “Service axes” display in the “Diagnosis” operating area.

**Determine example of error curve**

Installation of the temperature compensation is described below using the example of a Z axis on a lathe.

In order to determine the temperature-dependent error characteristic of the Z axis, proceed as follows:

- Constant heating 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 distances of 100 mm
- Measuring the actual temperature at the leadscrew
- Executing a traversing measuring cycle every 20 minutes

The mathematical and technological relationship 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 figure below.

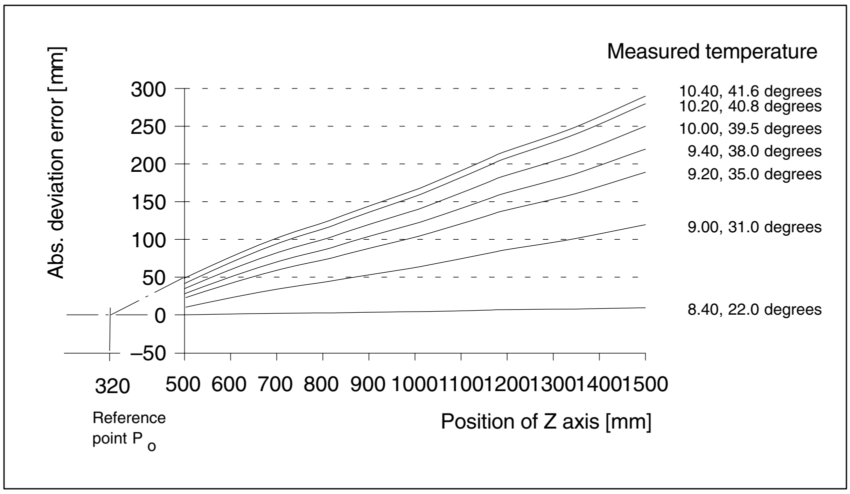



Fig. 2-3 Error curves determined for the Z axis

**Specifying parameters**

The temperature compensation parameters must now be set on the basis of the measurement results (see figure above).

**Reference position $P_0$**

As the figure above illustrates, there are basically two methods of configuring 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 our example, variant 2 is chosen, where the position-independent temperature compensation value is always 0. The temperature compensation value therefore only consists of the position-dependent components. The following parameters result:

- MD32750 $MA_TEMP_COMP_TYPE = 2 (only position-dependent temperature compensation active)
- $P_0$ = 320 mm => SD43920 $SA_TEMP_COMP_REF_POSITION = 320

**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 (see figure below).

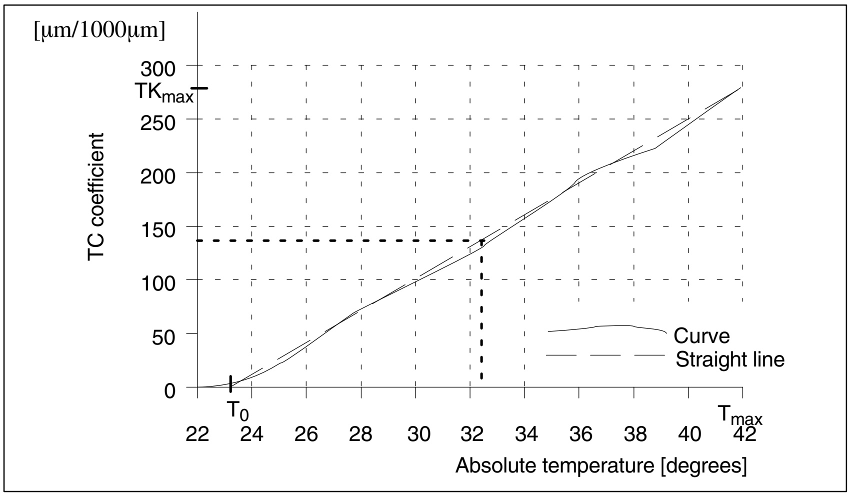



Fig. 2-4 Characteristic of coefficient tanβ as a function of measured temperature T

Depending on the resulting line, the following dependency on T results for the coefficient tanβ:

$$\tan\beta\ (T) = (T - T_0) * \frac{TK_{max}}{T_{max} - T_0}$$

where $T_0$ = temperature at which the position-dependent error = 0
$T_{max}$ = maximum measured temperature
$TK_{max}$ = Temperature coefficient at $T_{max}$

The results are as follows with reference to the figure above:

for $T_0$ = 23 degrees
$T_{max}$ = 42 degrees
$TK_{max}$ = 270 μm/1000 mm

tan ß(T) is therefore:

tanβ (T) = (T – 23 degrees) *14.21 [μm/1000 mm]
Example: for T = 32.3 degrees —> tanβ = 132μm/1000 mm

The coefficient tanβ (T) for every measured temperature T can be calculated easily in the PLC according to the above equation and then transferred to the NCK as SD 43910: TEMP_COMP_SLOPE.

## 2.2 Backlash compensation

**Mechanical backlash**

Slight backlash generally occurs in the power train between a moving machine part and its drive (e.g. leadscrew) since an unacceptably high level of machine wear would occur if the mechanical components were to be set to be absolutely free of backlash.
Backlash can also occur between the machine part and the measuring system.

**Effect**

In the case of axes/spindles with indirect measuring systems, mechanical backlash results in corruption of the traverse path, causing an axis, for example, to travel too much or too little by the amount of the backlash when the direction of movement is reversed (see the following 2 diagrams).

**Compensation**

To compensate for backlash, the axis-specific actual value is corrected by the amount of backlash every time the axis/spindle changes direction.

This quantity can be entered for each axis/spindle at the start-up phase in machine data MD 32450: BACKLASH. If there is a second measuring system installed for the axis/spindle, the relevant backlash values must be entered for each measuring system.

In SW 5 and later, the backlash can be weighted by a factor as a parameter set function. The weighting factor is set in MD 32452: BACKLASH_FACTOR to between 0.01 and 100.0, default setting is 1.0.
Application: e.g. compensation of gear-stage-related backlash.

**Activation**

Backlash compensation is always active in all operating modes after reference point approach.

**Position display**

The normal actual value and setpoint position displays ignore the compensation values and show the position values of an “ideal machine”.

**Display of compensation values**

The compensation value applying to the current actual position is output as the total compensation calculated from “LEC” and “backlash compensation” in the “Service axes” display in the “Diagnosis” operating area.

**Positive backlash**

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 (see figure below). The backlash compensation value must be entered as a **positive** value here (= normal case).

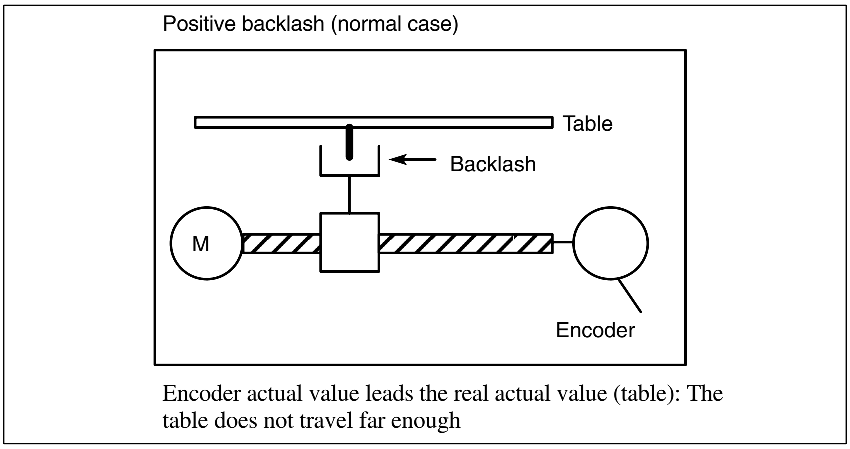



Fig. 2-5 Positive backlash (normal case)

**Negative backlash**

The encoder "lags behind" the machine part (e.g. table); the table then travels too far (see figure below). The correction value entered is **negative**.

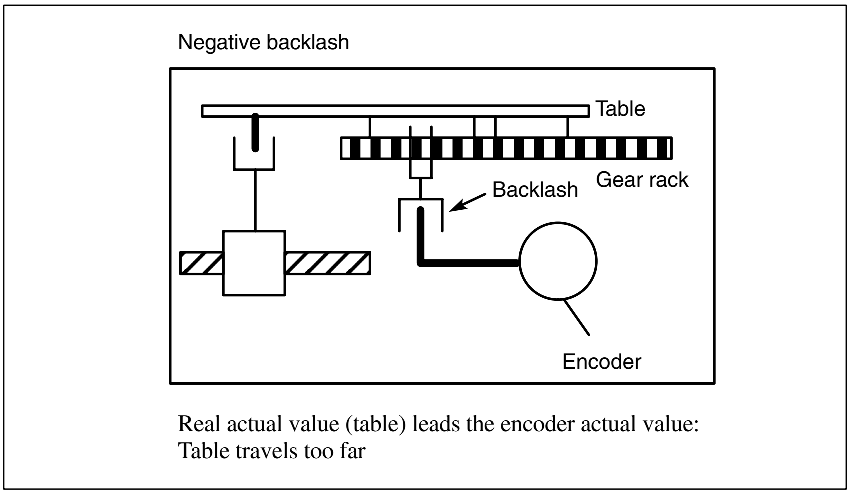



Fig. 2-6 Negative backlash

**2nd measuring system**

If there is a second 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 activated.

**Large backlash compensation**

The user has the option of applying the backlash compensation value gradually in several increments when the relevant axis reverses direction. This prevents an excessive setpoint step change from causing specific axis errors.

The contents of the axis-specific MD 36500: ENC_CHANGE_TOL determines the increment with which the backlash compensation value (MD 32450: BACKLASH) is applied.

Please note that the backlash compensation is fully calculated only after n (n = MD 32450 / MD 36500) servo cycles. An excessive time span can cause the triggering of zero speed monitoring alarms.

If MD 36500: ENC_CHANGE_TOL is set higher than
MD 32450: BACKLASH, the compensation is performed in one servo cycle.

# 2.3 Interpolatory compensation

## 2.3.1 General

**Compensation methods**

The following compensation methods are applied in order to implement “interpolatory compensation”:

1. “Leadscrew error compensation” or “measuring system error compensation” (referred to as **MSEC** in the following).
2. Beam sag compensation or angularity error compensation (Software Version 2 and higher), which from now on will be referred to as **beam sag compensation**.

Many of the characteristics of these two compensation methods are identical and are therefore described in the next Section “General notes”.

**Terms**

The following terms are used in the description of “Interpolatory compensation”:

| | |
|---|---|
| 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 or actual value position forms the basis of the calculation of a compensation value. |
| Compensation axis | Axis whose setpoint or actual value position is modified by a compensation value. |
| Interpolation point | A position of the base axis and the corresponding compensation value of the compensation axis. |
| Correction table | Table containing interpolation points. |
| Compensation relations | Assignment of the base axis and the corresponding compensation axis and the reference to the corresponding compensation table. |

**Leadscrew and measuring system errors**

The measuring principle of “indirect measurement” on NC-controlled machines is based on the assumption that the leadscrew pitch is constant at any given point within the traversing range so that the actual axis position 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).

To these are added the deviations caused by the measuring system used (differing divisions) and by the way the measuring system is mounted on the machine (measuring system errors) and any machine-dependent sources of error.

**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 (see Fig. 3.2).

Large workpieces, too, 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 table**

Since the deviations in dimension caused by the phenomena described above have a direct effect on workpiece machining accuracy, they need to be compensated by appropriate position-dependent correction values. The compensation values are derived from measured error curves and entered in the control in the form of compensation tables during installation. A separate table must be created for each compensation relation.

The compensation values and additional table parameters are entered in the compensation tables using special system variables.

---

**Note**

Compensation tables can be loaded only if MD 32700: ENC_COMP_ENABLE (interpolatory compensation)=0 and/or MD 32710: CEC_ENABLE (enable beam sag compensation) are set to zero.

---

**Input of compensation table**

The size of the compensation table, i.e. the number of interpolation points, must first be defined in a machine data – a power ON must then be executed.

Compensation tables can be loaded to the backed up NC user memory by two different methods.

- The compensation values are loaded when an NC program with the compensation tables is started.
- The compensation values can also be loaded by transferring the tables from a PC via the serial interface on the MMC.

---

**Note**

Once the size of the compensation tables has been defined in machine data, the NC generates the tables after the next power ON. The default setting for these tables is “0”.

The compensation tables can be output from the “Services” operating area via the serial interface on the MMC and loaded back after editing.

---

These compensation values are not lost when the control is switched off because they are stored in the non-volatile user memory. They can be updated if necessary (e.g. as a result of re-measuring because of machine aging).

**Caution**

When the setting in MD 18342: MM_CEC_MAX_POINTS[t] (max. number of interpolation points of beam sag comp., SRAM) or MD 38000: MM_ENC_COMP_MAX_POINTS (number of interpolation points for interpolatory comp., SRAM) is changed, the buffered NC user memory is reinitialized when the system powers up. All user data of the battery-buffered user memory (e.g. drive and MMC machine data, tool offsets, parts programs, compensation tables etc.) are deleted.

**References:** /FB/, S7, “Memory Configuration”

**Archiving**

Compensation tables are not saved with the series start-up file.

To archive compensation tables, they must be output via the serial interface on the MMC. The following compensation types can be selected for archiving in the operating area “Services”, “Data OUT”:

- LEC/measuring system error compensation (%_N_AX_EEC_INI)
- Beam sag/angularity compensation (%_N_AX_CEC_INI)
- Quadrant error compensation (%_N_AX_QEC_INI)

Compensation tables can also be saved as an archive file with an MMC102/103.

**Linear interpolation between the 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 figure below). The actual positions that limit these sub-paths are designated “interpolation points”. A compensation value must be entered for each interpolation point (actual position) during installation. The compensation value applied between 2 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).

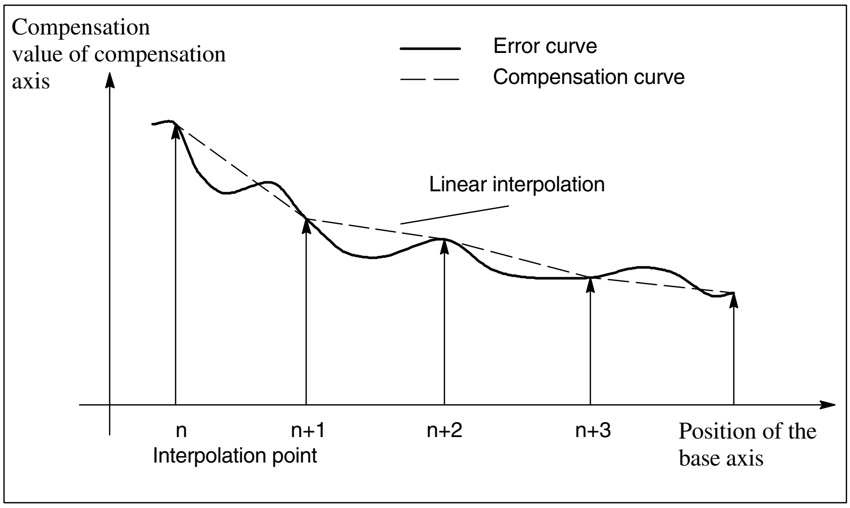



Fig. 2-7 Linear interpolation between the interpolation points

**Compensation value at reference point**

The compensation table should be structured such that the compensation value at the reference point is “zero”.

## 2.3.2 Measuring system error compensation (MSEC)

**Function**

The leadscrew error compensation function is part of the measuring system error compensation system.
In “Measuring system error compensation” (from now on referred to 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.

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, velocity 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.

**Activation**

The “MSEC” does not become active until the following conditions are fulfilled:

- The compensation values are stored in the NC user memory and active (after power ON).
- The function has been activated for the machine axis concerned (MD32700: ENC_COMP_ENABLE [e] = 1). If a 2nd position measuring system is to be compensated, this must also be enabled with the above machine data (e = 0: 1. measuring system; e = 1: 2. measuring system).
- The axis has been referenced (IS: “Referenced/synchronized 1 or 2” DB31, ... DBX60.4 or 60.5 = ‘1’).

As soon as these conditions have been fulfilled, the axis-specific actual value is altered 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 (IS “Referenced/synchronized 1 or 2”=‘0’), compensation processing is deactivated.

**Compensation interpolation points**

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 in MD 38000: MM_ENC_COMP_MAX_POINTS.

MD 38000: MM_ENC_COMP_MAX_POINTS[e,AXi]

where: AXi = axis name e.g. X1, Y1, Z1

e = measuring system (e = 0: 1. measuring system; e = 1: 2. measuring system)

$$\text{MM\_ENC\_COMP\_MAX\_POINTS[e, AXi]} = \frac{\text{\$AA\_ENC\_COMP\_MAX[e, AXi]} - \text{\$AA\_ENC\_COMP\_MIN[e, AXi]}}{\text{\$AA\_ENC\_COMP\_STEP[e, AXi]}} + 1$$

**Compensation table**

The position-related compensation values are stored in the form of system variables for the relevant axis in the compensation table.

The following measuring-system-specific parameters must be set for the table (see figure below):

- **Compensation value for interpolation point N in compensation table** ($AA_ENC_COMP [e,N,AXi])

  For every individual interpolation point (axis position) the compensation value must be entered in the table.

  Interpolation point N is limited by the number of possible interpolation points in the relevant compensation table (MD 38000: MM_ENC_COMP_MAX_POINTS).

  The magnitude of the compensation value is not limited.
  Permissible range of N: $0 \leq N <$ MM_ENC_COMP_MAX_POINTS –1

---

**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.

---

- **Distance between interpolation points** ($AA_ENC_COMP_STEP[e,AXi])
The distance between interpolation points corresponds to the distance between the compensation values in the relevant compensation table (see above for meaning of e and AXi).

- **Initial position** ($AA_ENC_COMP_MIN[e,AXi])
The initial position is the axis position at which the compensation table for the relevant axis begins ($\doteq$ interpolation point 0).

  The compensation value for the initial position is $AA_ENC_COMP_STEP[e,**0**,AXi)].

  For all positions smaller than the initial position the compensation value of interpolation point zero is used (does not apply for table with modulo).

- **End position** ($AA_ENC_COMP_MAX[e,AXi])
The end position is the axis position at which the compensation table for the relevant axis ends ($\doteq$interpolation point k).

  The compensation value for the end position is $AA_ENC_COMP_STEP[e,**k**,AXi)].

  The compensation value of interpolation point k is used for all positions larger than the end position (exception: table with modulo functions).

  The number of required interpolation points is calculated as follows:

$$k = \frac{\$AA\_ENC\_COMP\_MAX - \$AA\_ENC\_COMP\_MIN}{\$AA\_ENC\_COMP}$$

  With $0 \preceq k <$ MD 38000: MM_ENC_COMP_MAX_POINTS

  The following conditions apply to interpolation point k:

  – With k = MD 38000: MM_ENC_COMP_MAX_POINTS – 1

    ⇒ the compensation table is fully utilized!

  – With k < MD 38000: MM_ENC_COMP_MAX_POINTS – 1

    ⇒ the compensation table is not fully utilized; compensation values entered in the table greater than k have no effect.

- With k > MD 38000: MM_ENC_COMP_MAX_POINTS – 1

  ⇒ the compensation table is limited internally by reducing the end position; the compensation values greater than k are not used.

- **Compensation with modulo function**
($AA_ENC_COMP_IS_MODULO[e,AXi])
When the compensation is activated with a modulo function, the compensation table is repeated cyclically, i.e. the compensation value at location $AA_ENC_COMP_MAX ($\doteq$interpolation point $AA_ENC_COMP[e,k,AXi]) is followed immediately by the compensation value at location $AA_ENC_COMP_MIN ($\doteq$interpolation point $AA_ENC_COMP[e,0,AXi]).

  For rotary axes with modulo 360° it is therefore suitable to program 0° as the initial position ($AA_ENC_COMP_MIN) and 360° as the end position ($AA_ENC_COMP_MAX).

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.

| | |
|---|---|
| $AA_ENC_COMP_IS_MODULO[e,AXi] = 0: | Compensation without modulo function |
| $AA_ENC_COMP_IS_MODULO[e,AXi] = 1: | Compensation with modulo function |

**Caution**

When the compensation values are entered it is important that all interpolation points be assigned a position value within the defined range (i.e. no gaps). Otherwise, the previous valid position value is used for these interpolation points.

**Note**

Table parameters which contain position information are not automatically converted at measuring system change (change in MD 10240: SCALING_SYSTEM_IS_METRIC) in SW 4 and lower. The position information is always interpreted in the current measuring system. Conversions must be conducted externally.

In SW 5 and higher, when MD 10260: CONVERT_SCALING_SYSTEM=1, it is possible to configure automatic conversion of position data. External conversion is no longer necessary.

**References:** /FB1/, G2, Chapter 2

**Example**

The following example shows compensation value inputs for machine axis X1.

```
%_N_AX_EEC_INI
CHANDATA (1)
$AA_ENC_COMP[0,0,X1]        = 0.0       ; 1st compensation value
                                          (≙ interpolation point 0) +0µm
$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)
$AA_ENC_COMP_STEP[0,X1] = 1.0           ; Distance between interpolation
                                          points 1.0 mm
$AA_ENC_COMP_MIN[0,X1]      = –200.0    ; Compensation begins
                                          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
```

In this example, the number of compensation interpolation points set in MD38000: MM_ENC_COMP_MAX_POINTS $\geq$ must be 801 or else alarm 12400 “Element does not exist” will be output.

The compensation table for this example requires at least 6.4KB of the non-volatile NC user memory (8 bytes per compensation value).

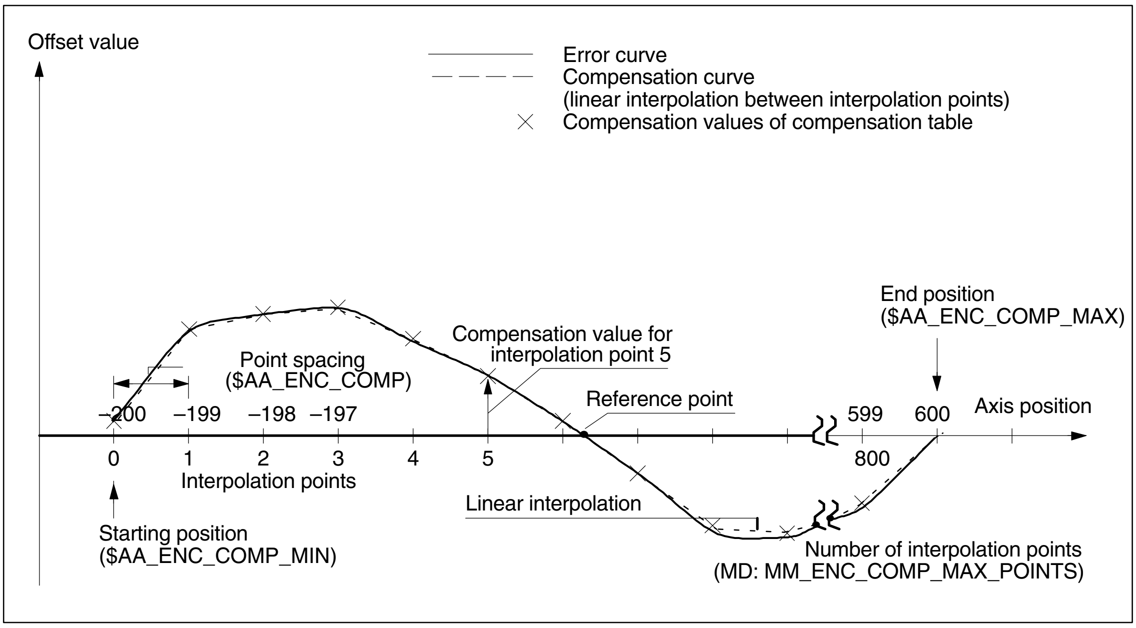



Fig. 2-8 Compensation table parameters (system variables for MSEC)

### 2.3.3 Sag compensation and angularity error compensation

**Function**

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 figure 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.

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.

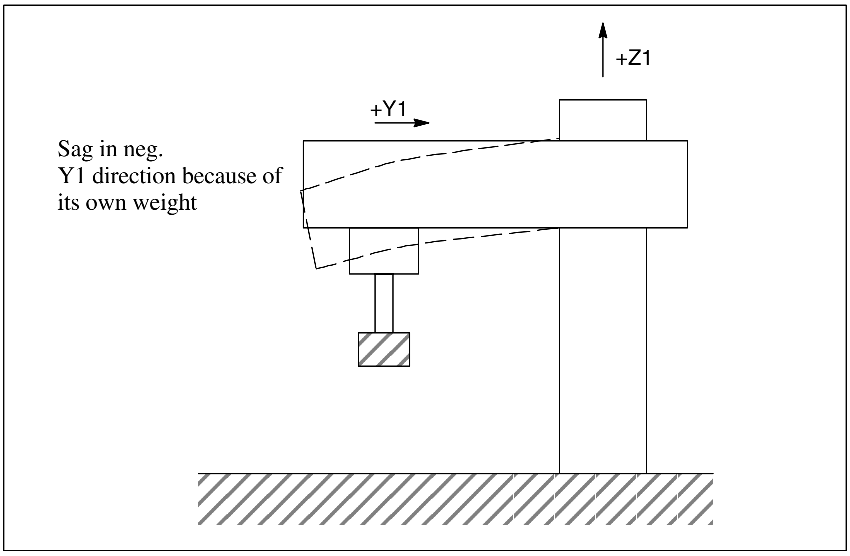



Fig. 2-9 Example of sag caused by own weight

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 figure 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 in system
   – Maximum number of tables applied to one compensation axis:
     1 * maximum number of axes in 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 range of influence 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 parts program, PLC or the user at any time) can be introduced for every compensation table.
8. Compensation tables can be multiplied in pairs (settable via system variables). The product is added to the total compensation value of the compensation axis.
9. The compensation can be activated in the following ways:
   – MD 32710: CEC_ENABLE [AXi] the sum of all compensation relations is enabled for machine axis AXi.
   – With MD 41300: $SN_CEC_TABLE_ENABLE[t], evaluation of the compensation table [t] is enabled.

     It is thus possible, for example, to alter the compensation relations either from the parts program or from the PLC user program (e.g. switching over the tables), depending on the machining requirements.
10. In SW 5 and higher, when MD 10260: CONVERT_SCALING_SYSTEM=1 is set, the axial MD 32711: CEC_SCALING_SYSTEM_METRIC becomes effective. 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.

---

**Note**

No compensation table becomes active until the base axis and compensation axis have been referenced.

---

**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 compensation value is set in axial MD 32720: CEC_MAX_SUM for specific axes.

If the resulting total compensation value is greater than the maximum value, alarm 20124 “Sum of compensation values too high” is output. Program processing is not interrupted. The compensation value output as an additional setpoint is limited to the maximum value.

Alteration of the total compensation value is also limited axially. When limit value MD 32730: CEC_MAX_VELO is exceeded, alarm 20125 “Compensation value changed too quickly” is output; again program processing is continued. The path not covered because of the limitation is made up as soon as the compensation value is no longer subject to limitation.

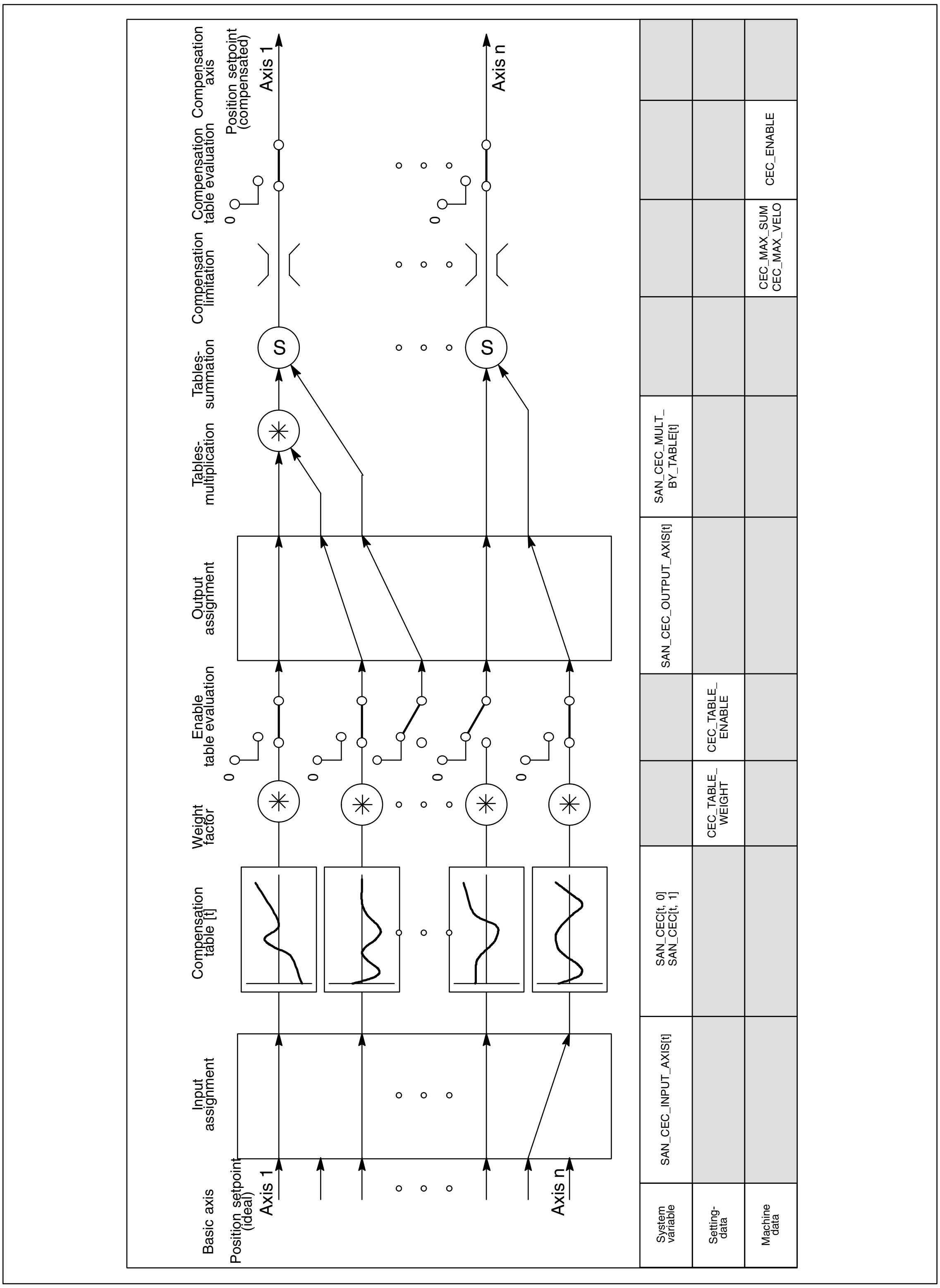



Fig. 2-10 Generation of compensation value for sag compensation

**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.

**Activation**

The beam sag compensation function does not become active until the following conditions are fulfilled:

- The option “Interpolatory compensation” has been enabled.
- The function has been activated for the relevant machine axis (compensation axis)
  (MD 32710: CEC_ENABLE [AXi] = 1).
- The compensation values have been stored in the non-volatile NC user memory and are active (after power ON).
- Evaluation of the relevant compensation table has been enabled
  (SD 41300: CEC_TABLE _ENABLE [t] = 1)
- The current measuring system of the base and compensation axes has been referenced (IS: “Referenced/synchronized 1 or 2” DB31, ... DBX60.4 or 60.5 = ‘1’).

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 (IS “Referenced/Synchronized 1 or 2” = ‘0’), compensation processing is deactivated.

**Compensation interpolation points**

The number of required interpolation points in the compensation table must be defined for every compensation relationship and the requisite memory space reserved in general MD 18342: MM_CEC_MAX_POINTS.

MD 18342: MM_CEC_MAX_ POINTS[t]

where: [t] = Index of compensation table
with (0 ≤ t < 2 * maximum number of axes)
i.e. t = 0: 1. compensation table
t = 1: 2. compensation table etc.

$$MM\_CEC\_MAX\_POINTS[t] = \frac{\$AN\_CEC\_MAX[t] - \$AN\_CEC\_MIN[t]}{\$AN\_CEC\_STEP[t]} + 1$$

**Table parameters**

The position-related corrections for the relevant compensation relationship are stored as system variables in the compensation table.

The following parameters must be set for the table (see Fig. 3.1):

- **Compensation value for interpolation point N in compensation table [t]** ($AN_CEC [t, N])

  The compensation value of the compensation axis must be entered in the table for each individual interpolation point (position of the base axis).

  Interpolation point N is limited by the number of possible interpolation points in the relevant compensation table (MD 18342: MM_CEC_MAX_POINTS).

  Permissible limit of N: $0 \leq N <$ MD 18342: MM_CEC_MAX_POINTS

- **Base axis** ($AN_CEC_INPUT_AXIS[t])
  Name of machine axis whose setpoint is to be used as the input for the compensation table [t].

- **Compensation axis** ($AN_CEC_OUTPUT_AXIS[t])
  Name of machine axis to which the output of the compensation table [t] is to be applied.

---

**Note**

In multi-channel systems the "general axis identifiers" AX1... must be preset, if the identifiers of machine axis and channel axis are identical.

---

- **Distance between interpolation points** ($AN_CEC_STEP[t])
  The distance between interpolation points defines the distance between the input values for the compensation table [t].

- **Initial position** ($AN_CEC_MIN[t])
  The initial position is the position of the base axis at which the compensation table [t] begins ($\doteq$ 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).

- **End position** ($AN_CEC_MAX[t])
  The end position is the position of the base axis at which the compensation table [t] ends ($\doteq$ interpolation point k).

  The compensation value for the end position is $AN_CEC [t,**k**].

  The compensation value of interpolation point k is used for all positions larger than the end position (exception: table with modulo functions).

  The number of required interpolation points is calculated as follows:

  $$k = \frac{\$AN\_CEC\_MAX[t] - \$AN\_CEC\_MIN[t]}{\$AN\_CEC\_STEP[t]}$$

  With $0 \leq k <$ MD 18342: MM_CEC_MAX_POINTS

  The following conditions apply to interpolation point k:

  – With k = MD 18342: MM_CEC_MAX_POINTS – 1
    ⇒ the compensation table is fully utilized!

– With k < MD 18342: MM_CEC_MAX_POINTS – 1

⇒ the compensation table is not fully utilized; the entered compensation values greater than k have no effect

– With k > MD 18342: MM_CEC_MAX_POINTS – 1

⇒ the compensation table is limited in the control by reducing the end position; the compensation values greater than k are not used

- **Direction-dependent compensation** ($AN_CEC_DIRECTION[t])
This system variable can be used to define whether the compensation table [t] should apply to both travel directions of the base axis or only either the positive or negative direction.
 0: Table affects both traversing directions of the base axis
 1: Table only affects the positive traversing direction of the base axis
 –1: Table only affects the negative traversing direction of the base 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.

- **Table multiplication** ($AN_CEC_MULT_BY_TABLE[t])
This option allows the compensation values of any table to be multiplied with those of another (or with themselves). The product is added as an additional compensation value to the total compensation value of the compensation table.
Syntax: $AN_CEC_MULT_BY_TABLE[$t_1$] = $t_2$
 $t_1$ = Index of table 1 of the compensation axis
 $t_2$ = Number of table 2 of the compensation axis
 It is important to ensure that the number and index of the **same table** are different!
 The general rule is: Table number = table index + 1

- **Compensation with modulo function** ($AN_CEC_IS_MODULO[t])
When the compensation with modulo function is activated, the compensation table is repeated cyclically, i.e. the compensation value at location $AN_CEC_MAX[t] (interpolation point $AN_CEC[t,k]) is followed immediately by the compensation value at location $AN_CEC_MIN[t] (interpolation point $AN_CEC[t,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.
$AN_CEC_IS_MODULO[t]= 0: Compensation without modulo function
$AN_CEC_IS_MODULO[t]= 1: Compensation with modulo function
If modulo compensation is to be implemented with a modulo rotary axis as base axis, the compensation table used has to be modulo calculated as well.
Example:
MD 30300: IS_ROT_AX[AX1] = 1: Rotary axis
MD 30310: 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

**Note**

Table parameters which contain position information are not automatically converted at measuring system change (change in MD 10240: SCALING_SYSTEM_IS_METRIC) in SW 4 and lower. The position information is always interpreted in the current measuring system. Conversions must be conducted externally.

In SW 5 and higher, when MD 10260: CONVERT_SCALING_SYSTEM=1 is set, the measuring system can be configured via axial MD 32711: CEC_SCALING_SYSTEM. External conversions of position information are no longer necessary with a measuring system change.

**References:** /FB1/, G2, Chapter 2

**Table example**

The following example shows the compensation table for sag compensation of axis Y1. Depending on the position of the Y1 axis, a compensation value is applied to the Z1 axis. The 1st compensation table (t=0) is used for this.

```
%_N_NC_CEC_INI
CHANDATA(1)
$AN_CEC [0,0]               = 0       ; 1st compensation value (≙ inter-
                                        polation point 0) for Z1: ±0µm
$AN_CEC [0,1]               = 0.01    ; 2nd compensation value (≙ inter-
                                        polation point 1) for Z1: +10µm
$AN_CEC [0,2]               = 0.012   ; 3rd compensation value ≙ inter-
                                        polation point 2) for Z1: +12µm
:
$AN_CEC [0.100]             = 0       ; Last compensation value ≙ inter-
                                        polation point 101) for Z1: ±0µm
$AN_CEC_INPUT_AXIS[0]       = (AX2)   ; Base 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 begins at
                                        Y1 = –400 mm
$AN_CEC_MAX[0]              = 400.0   ; Compensation begins at
                                        Y1 = +400 mm
$AN_CEC_DIRECTION[0]        = 0       ; Table applies to both directions of
                                        travel of Y1
$AN_CEC_MULT_BY_TABLE[0] =            ;
$AN_CEC_IS_MODULO[0]        = 0       ; Compensation without modulo
                                        function
M17
```

In this example, the number of compensation interpolation points set in MD18342: MM_CEC_MAX_POINTS [0] must be ≥ 101; otherwise alarm 12400 is activated.

The compensation table for this example requires at least 808 bytes of non-volatile NC user memory.

**Table multiplication**

With the table multiplication function, any table can be multiplied with any other table (i.e. even with itself). The multiplication link is established using the system variables described above.

The following example for the compensation of machine foundation sagging illustrates an application of table multiplication.

On large machines, sagging of the foundation can cause inclination of the whole machine. For the boring mill in the second figure below, for example, it is determined that compensation of the X1 axis is dependent both on the position of the X1 axis itself (since this determines angle of inclination $\beta$) and on the height of the boring mill (i.e. the position of the Z1 axis).

To implement compensation, the compensation values of the X1 and Z1 axes must be multiplied according to the following equation (see figure below):

$$\Delta X1 = Z1 * \sin\beta(X1) \approx Z1 * \beta(X1)$$

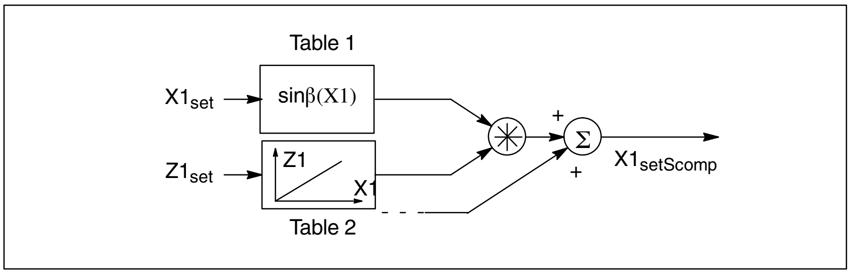



Fig. 2-11 Table multiplication

Compensation table 1 (table index = 0) describes the reaction of axis X1 on axis X1 (sine of the position-dependent tilting angle $\beta$(X1)).

Compensation table 2 (table 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

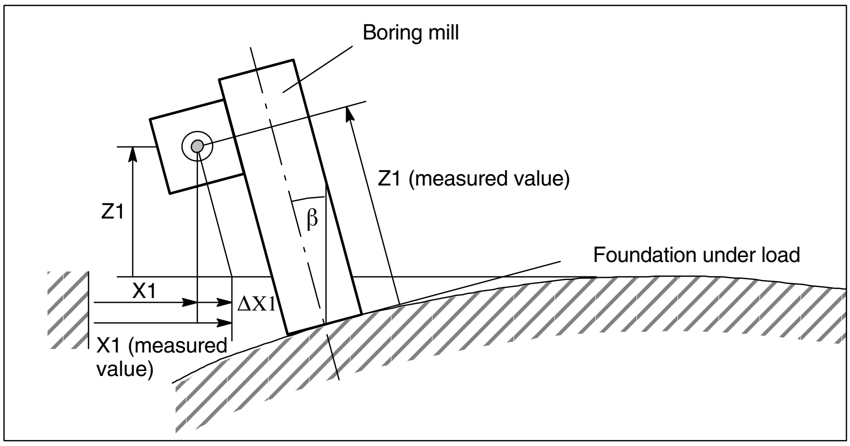



Fig. 2-12 Compensation of sag in a machine base

**Example: 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. Where such conditions need to be met, 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 on 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 2000x900mm$^2$. The compensation values are each measured in steps of 500mm along the x axis and 300mm along the y axis.

---

**Note**

The maximum dimensions of the grid (number of lines and columns) depends on the following points:

No. of lines: Dependent on number of axes in the system (dependent on NCU type)

No. of columns: Dependent on the maximum number of values which can be entered in a compensation table (up to a total of 2000 values)

---

**Caution**

The number of lines and columns is set in MD 18 342: MM_CEC_MAX_POINTS). The machine data is memory-configuring.

---

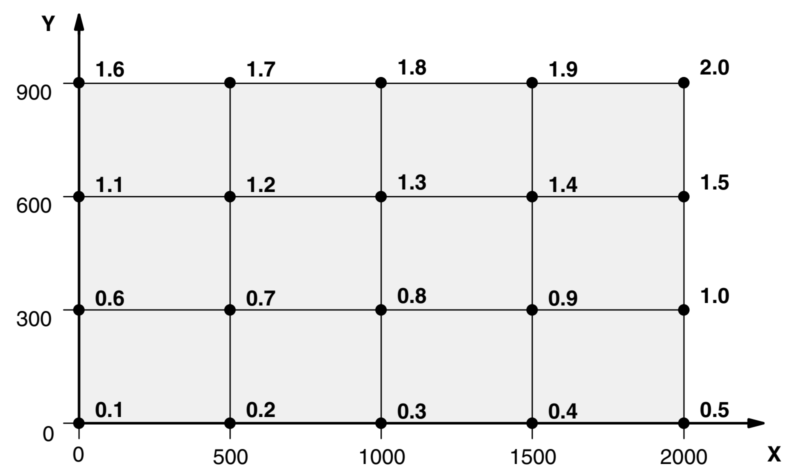



Fig. 2-13 Compensation values of Z axis with chessboard-like distribution of X-Y plane

**Fundamental principle**

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). Compensation values 0.1 to 0.5 are entered in the first table in the example and compensation values 0.6 to 1.0 in the second (see Fig. 2-13). Compensation tables are referred to below as f tables and the table values as f_i(x) (i = number of table).

The compensation values of f tables are evaluated by multiplying them with 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 with f tables, the correct compensation value in each f table is selected by multiplying it with 1. All irrelevant compensation values are concealed through multiplication with 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 current 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 with 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

(for functions values, see also f and g tables in program code)

**Program code**

The application example described above can be achieved with the following parts program code:

```
$MA_CEC_ENABLE[Z1]        = FALSE  ; Deactivate the compensation by
                                     setting to FALSE, allowing the table
                                     values to be altered without gener-
                                     ating 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 take effect after power ON.
;Define base 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
;with the number [t2]
;in accordance with the rule of calculation specified above
$SN_CEC_MULT_BY_TABLE_[0]=5
$SN_CEC_MULT_BY_TABLE_[1]=6
$SN_CEC_MULT_BY_TABLE_[2]=7
$SN_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 take effect after 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     ; Reactivate compensation
NEWCONF

;Execute a program test to check effectiveness of compensation.
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 MMC. To do so, select soft key “Diagnosis” followed by softkey “Service axis”. The currently effective compensation value is displayed next to the “Sag + temperature compensation” variable.

---

```
;The power ON machine data are set to prepare the table configuration
;cec.md:
;Set option data for start-up

;Define the number of interpolation points in the compensation tables
;Machine data is memory-configuring
$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 max. total compensation
                                  value
$MA_CEC_MAX_VELO[AX3]=100.0     ; Limit the max. changes to the total
                                  compensation value
M17
```

## 2.3.4 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).

In the axis/spindle service display (operating area Diagnosis) the positional value determined by the measuring system plus the sum of the backlash compensation and leadscrew error compensation is displayed (= actual position value measuring system 1/2).

**Compensation value display**

The following compensation values are also output in the "Axes" service display (Diagnosis operating area):

| Axes service display | Meaning |
|---|---|
| Absolute compensation value measuring system 1 or 2 | Display value corresponds to the total compensation value calculated from "MSEC" and "Backlash compensation" for the current actual position of the axis (measuring system 1 or 2). |
| Compensation value beam sag/temperature | Display value is the sum of the compensation values from "beam sag compensation" and "temperature compensation" for the current actual position of the axis. |

**References:** /FB/, D1, "Diagnostics Tools"

**Reference point loss**

If the reference point for the base axis is lost (IS: "Referenced/synchronized 1 or 2" DB31, ... DBX60.4 or 60.5 = '0'), the MSEC and sag compensation functions are deactivated in the affected axes. When the reference point is reached these compensations are automatically switched on again.

**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.

**Output of travel signals**

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.

## 2.4 Dynamic feedforward control (following error compensation)

### 2.4.1 General

**Axial following errors**

The axial following error can be reduced almost to zero with the help of the feedforward control. This feedforward control is therefore also called “following error compensation”.

The following error causes undesired velocity-dependent contour errors especially during acceleration at contour curves, e.g. arcs and corners.

**Feedforward control methods**

The following feedforward control methods can be used to implement “following error compensation”:

- **Speed feedforward control** (velocity dependent)
  linked to SINUMERIK 840D (part of the Basic Version for SINUMERIK 840D)
- **Torque feedforward control** (acceleration dependent)
  linked to SIMODRIVE 611 digital (option for SINUMERIK 840D)

---

**Note**

The torque type of feedforward control is not supported by the SIMODRIVE 611 universal drive with the SINUMERIK 840Di or SINUMERIK 840D with PROFIBUS DP.

---

**Activation**

MD 32620: FFW_MODE must first be set to select the desired feedforward control mode.

0 = No feedforward control

1 = Speed feedforward control (default setting)

2 = Torque feedforward control (only possible for SINUMERIK 840D in conjunction with drives supporting this function, e.g. SIMODRIVE 611 digital)
The option must be enabled before selecting torque feedforward control.

3= Speed feedforward control with Tt balancing for SW 5.1 and higher

4= Torque feedforward control (only possible with SINUMERIK 840D) with Tt balancing for SW 5.1 and higher
The option must be enabled before selecting torque feedforward control.

**Setting the type of feedforward control**

MD 32630: FFW_ACTIVATION_MODE defines for each axis whether the feedforward control is to be selected according to the status of MD 32620: FFW_MODE or whether it can also be activated from the part program. The feedforward control type is selected from MD 32620: FFW_MODE if

MD 32630: FFW_ACTIVATION_MODE = 0.

The feedforward control type can be selected within the part program if

MD 32630: FFW_ACTIVATION_MODE = 1.

**Upgrading 840Di to SW 6.3**

When upgrading the SINUMERIK 840Di to software version SW 6.3 or higher, the start-up settings must be reconfigured.

---

!

**Important**

If the feedforward control variant MD 32620: FFW_MODE = 3 has already been used **on an 840Di**, the startup setting in MD 32810: EQUIV_SPEEDCTRL_TIME must be reconfigured on a software upgrade to version 6.3, because Ti and To are set automatically. These settings must be corrected manually in MD 32810: EQUIV_SPEEDCTRL_TIME.

---

**Activation/deactivation in part program**

The feedforward control can also be activated and deactivated by means of the following high-level language elements in the part program:

FFWON Feedforward control ON
FFWOF Feedforward control OFF

The default setting (i.e. M30 even after Reset) is entered in channel-specific MD20150: GCODE_RESET_VALUES (initial setting of G groups).

FFWON and FFWOF are used to activate and deactivate respectively the feedforward control of all axes/spindles in the channel for which MD 32630: FFW_ACTIVATION_MODE = 1 is set (as well as MD 32620: FFW_MODE = 1 or 2).

MD 32630: FFW_ACTIVATION_MODE should therefore have identical settings for axes that interpolate with each other.

The feedforward control should only be switched on or off while the axis/spindle is stationary to prevent jerk. This is the responsibility of the programmer.

**Conditions**

The following points should be noted before the feedforward control is applied:

- Rigid machine behavior
- Precise knowledge about the machine dynamic response
- No sudden changes in the position and speed setpoints

**Optimization of control loop**

The feedforward control is set on an axis/spindle-specific basis. First of all, the current control loop, speed control loop and position control loop must be set to an optimum for the axis/spindle.

**References:** /IAD/, SINUMERIK 840D Installation & Start-Up Guide

**Parameter assignments**

The feedforward control parameters must then be assigned to the relevant axis/spindle and then entered in the machine data.

## 2.4.2 Speed feedforward control

In the case of speed feedforward control, a velocity setpoint is also applied directly to the input of the speed controller (see figure below). This additional setpoint can be weighted by a factor that must equal approximately 1 as standard.

In order to achieve a correctly set speed feedforward control, the equivalent time constant of the speed control loop must be determined exactly and entered as a machine data.

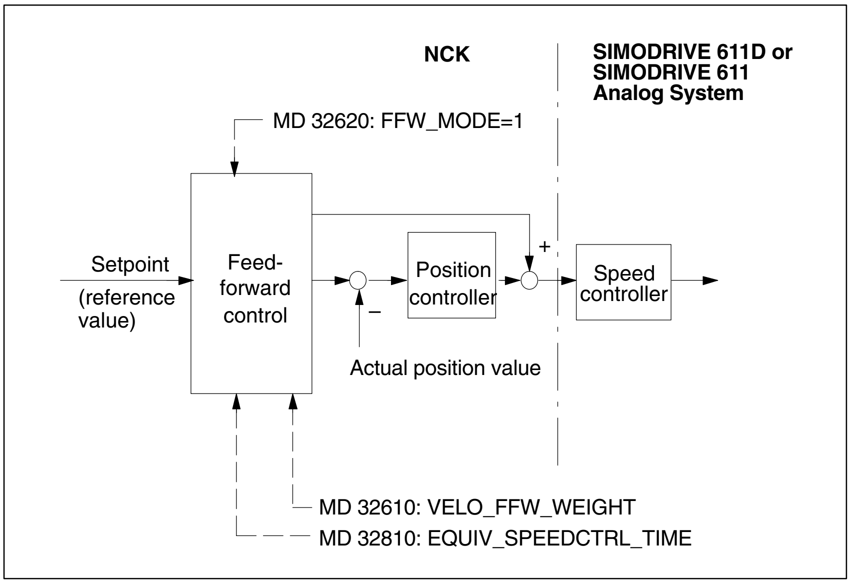



Fig. 2-14 Speed feedforward control

**Parameters**

The following axis-specific parameters must be defined for the speed feedforward control during installation:

- MD 32610: VELO_FFW_WEIGHT
  Feedforward control factor
- MD 32810: EQUIV_SPEEDCTRL_TIME
  Equivalent time constant of the closed speed control loop.

**Parameter for speed feed-forward control**

**MD 32810: EQUIV_SPEEDCTRL_TIME**
Equivalent time constant of the closed speed control loop.

The equivalent time constant of the closed speed control loop is determined by measuring the step response of the speed control loop. With the 611D, the settling process can be displayed using the installation tools.

**References:** /IAF/, "Installation and Start-Up Guide"
/IAD/, "Installation and Start-Up Guide"

The equivalent time constant of the speed control loop can also be generated from the position control cycle (=basic system cycle x factor for position control cycle) plus the speed setpoint filter (drive machine data 1500 ... 1521).

**MD 32610: VELO_FFW_WEIGHT**
Feedforward control factor for speed feedforward control

If the control loop for axis/spindle is optimally set and the equivalent time constant has been determined exactly, the feedforward control factor will be approximately **1**. Therefore the initial value to be entered in the machine data is 1 (= standard default setting).

With this value the following error will be reduced to nearly zero (i.e. control deviation is 0) when speed is constant. This should be checked by making positioning movements based on the actual “control deviation” shown on the service display.

**References:** /FB/, D1, “Diagnostics Tools”

**Fine adjustment of MD 32810**

By making fine adjustments to the values set in MD 32610: VELO_FFW_WEIGHT and MD 32810: EQUIV_SPEEDCTRL_TIME, it is possible to set the desired response for the relevant axis/spindle.

This is done by traversing the axis/spindle at a constant velocity and checking the affect of the changes made in the machine data in the service display Control deviation. The adjustment criterion for the speed feedforward control is ’control deviation’ = 0.

Case 1: When the axis is traversed in the positive direction the ’control deviation’ displays a **positive** value.
⇒ The equivalent time constant of the speed control loop or the feedforward factor is **too small**

Case 2: When the axis is traversed in the positive direction the ’control deviation’ displays a **negative** value.
⇒ The equivalent time constant of the speed control loop or the feedforward factor is **too large**

A small acceleration and a large 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.

In SW 5 and higher, you can optimize the position setpoint with a second balancing filter.

**References:** /FB/, G2, “Velocities, Setpoint/Actual Value Systems”, “Optimizing the control”

## Examples

Example with X axis:

MD 32300: MAX_AX_ACCEL = 0.1 ; m/s$^2$
MD 32000: MAX_AX_VELO = 20000.0 ; mm/min

; Part program for setting the equivalent time constant

```
G1 F20000
FFWON
LOOP:
X1000
X0
GOTOB LOOP
M30
```

Example for active speed feedforward control of axes 1, 2 and 3:

Equivalent time constant of the speed control loop
(MD 32810: EQUIV_SPEEDCTRL_TIME) for

- Axis 1: 2 msec
- Axis 2: 4 msec (dynamically the slowest axis)
- Axis 3: 1 msec

The values for the time constant of the dynamic response adaptation (MD32910: DYN_MATCH_TIME) are then as follows for:

- Axis 1: 2 msec
- Axis 2: 0 msec
- Axis 3: 3 msec

**References:** /FB/, G2, “Velocities, Setpoint/Actual Value Systems, Closed-Loop Control”

**Lead time for the speed setpoint**

With machine data MD 10082 and MD 10083, the transfer of the speed setpoint to the drive is adjustable.

---

**Note**

It is only possible to fix the lead time for output of speed setpoints with the digital 611D drives.

---

MD 10082: CTRLOUT_LEAD_TIME. The larger the value entered, the sooner the drive transfers the speed setpoints.

The following meanings apply:

- 0 %: Setpoints are transferred at the beginning of the next position control cycle
- 50 %: Setpoints are already transferred after execution of half of the position control cycle

A reasonable lead time can only be determined by measuring the maximum position control calculating time. In the machine data 10083: CTRLOUT_LEAD_TIME_MAX suggests a value measured by the control. As this is a net value, it is advisable for the user to provide for a safety allowance of, for example, 5%.

---

**Note**

If lead times that are input are too high, this can cause output of drive alarm 300506.

---

The input value is rounded to the next lower speed controller pulse rate in the drive. If the speed controller pulse rates of the drives are different, changing the value will not necessarily lead to the same degree of controller improvement for all configured drives.

### 2.4.3 Torque feedforward control (not 840Di)

In the case of torque feedforward control, an additional current setpoint proportional to the torque is applied directly to the current controller input (see figure below). This value is formed using the acceleration and moment of inertia.

In order to achieve a correctly set torque feedforward control, the exact equivalent time constant must be determined and entered in the machine data.

Because of the direct current setpoint injection, torque feedforward control is only possible with digital drives (SINUMERIK 840D).

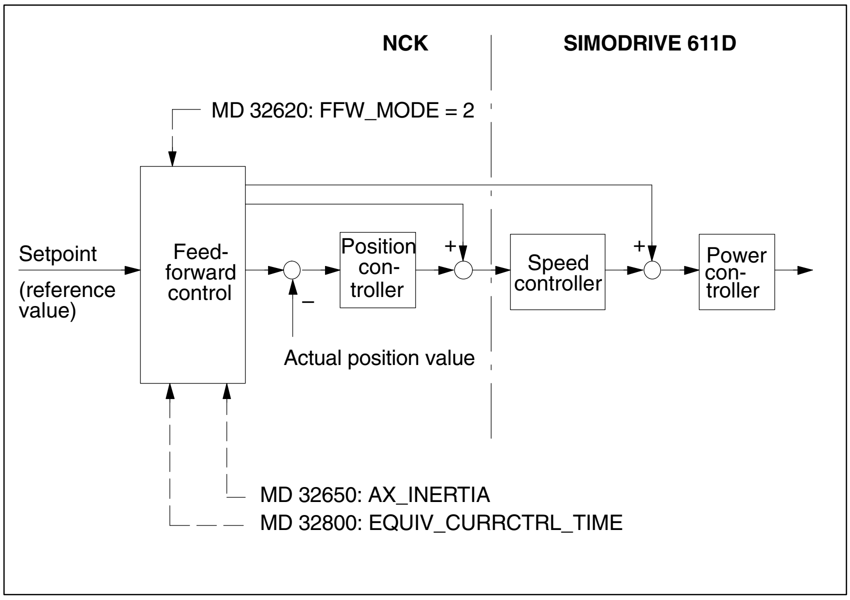



Fig. 2-15 Torque feedforward control

**Application**

Torque feedforward control is required to achieve high contour accuracy where the demands on the dynamics are great. If set correctly, the following error can almost be completely compensated even during high acceleration.

**Parameters**

The following axis-specific parameters must be defined during installation for torque feedforward control:

- MD 32650: AX_INERTIA
  Moment of inertia of the axis for torque feedforward control (from the point of view of the drive)
- MD 32800: EQUIV_CURRCTRL_TIME
  Equivalent time constant of current control loop
- SIMODRIVE 611D-MD 1004: CTRL_CONFIG
  Configuration structure
  Set bit 0 = “1” (torque feedforward control active)

**Parameters for torque feedforward control**

(available only on SINUMERIK 840D)

**SIMODRIVE 611D MD 1004: CTRL_CONFIG** Configuration structure

The torque feedforward control is activated in the SIMODRIVE 611D with bit 0 = 1.

**MD 32800: EQUIV_CURRCTRL_TIME**
Equivalent time constant of closed current control loop

The equivalent time constant of the closed current control loop is determined by measuring the step response of the current control loop.
With the SIMODRIVE 611D, the settling process can be displayed using the installation tool.

In addition, the current setpoint of the 1st drive of each module on the 1st DA converter of the module is output so that it can also be observed with an oscilloscope.

**References:** /IAD/, “Installation and StartUp Guide”

The equivalent time constant must be determined as exactly as possible.

**MD 32650: AX_INERTIA**
Total moment of inertia of the axis

The total moment of inertia (moment of inertia of drive + load referred to the motor shaft) of the axis must be determined and entered in the MD for torque feedforward control.

1 to 2 times the SIMODRIVE 611D-MD 1117: MOTOR_INERTIA (motor moment of inertia) is the recommended initial value setting for MD 32650: AX_INERTIA.

**Fine adjustment**

By making fine adjustments to the values set in MD 32800: EQUIV_CURRCTRL_TIME and MD 32650: AX_INERTIA it is possible to set the desired response for the relevant axis/spindle.
Because acceleration is very fast the service display cannot be used to finely adjust the parameters. In the case of the SIMODRIVE 611D, for example, changes made to the machine data should be checked by recording the following error from an analog setpoint output (this can only be done with the installation tool).

It is important to observe the following error against constant travel even when the axis/spindle is accelerating.
The adjustment criterion for torque feedforward control is:
Following error $\approx$ 0

Case 1: When the axis is traversed in the positive direction the following error shows a **positive** value.
$\Rightarrow$ The values entered for the equivalent time constant of the current control loop or for the moment of inertia of the axis are **too small**

Case 2: When the axis is traversed in the positive direction the following error shows a **negative** value.
$\Rightarrow$ The values entered for the equivalent time constant of the current control loop or for the moment of inertia of the axis are **too large**

**Setting for interpolating axes**

The feedforward control parameters must be set optimally for each axis even in the case of interpolating axes. The axes can have **different** feedforward control settings.

**Check contour monitoring**

The two equivalent time constants (MD 32810: EQUIV_SPEEDCTRL_TIME and MD 32800: EQUIV_CURRCTRL_TIME) influence the contour monitoring which should therefore subsequently be checked.

**References:** /FB/, A3, "Axis Monitoring Functions, Protection Zones"

**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 set in MD 32200: POS_CTRLGAIN has very little influence on the response to setpoint changes (e.g. corner errors, overshoots, circle/radius errors).

On the other hand, feedforward control does not affect the disturbance characteristic (synchronism). In this case, the factor set in MD 32200: POS_CTRLGAIN 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 applied to response to setpoint changes) shown in the Service display is very high.

**Dynamic response adaptation**

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.

When a feedforward control is active, the difference between the equivalent time constants of the "slowest" speed or current control loop for the relevant axis must be entered as the time constant of the dynamic response adaptation (MD 32910: DYN_MATCH_TIME).

In SW 5 and higher, you can optimize the position setpoint with a second balancing filter.

**References:** /FB/, G2, "Velocities, Setpoint/Actual Value Systems", "Optimizing the control"

# 2.5 Friction compensation (quadrant error compensation)

## 2.5.1 General

**Function**

Friction occurs predominantly in the gearing and guideways. Static friction is especially noticeable in the machine axes. Because it takes a greater force to initiate a movement (breakaway) than to continue it, a greater following error occurs at the beginning of a movement.

The same phenomenon occurs on a change of direction where static friction causes a jump in frictional force. If, for example, one axis is accelerated from a negative to a positive velocity, it sticks for a short time as the velocity passes through zero because of the changing friction conditions. With interpolating axes, changing friction conditions can cause contour errors.

**Quadrant errors**

This behavior is particularly apparent on circular contours on which one axis is moving at maximum path velocity and the other is stationary at quadrant transitions. With the aid of friction compensation these so-called "quadrant errors" can be almost completely eliminated.

**Principle**

Measurements on machines have shown that contour errors caused by static friction can be effectively compensated by the injection of an additional setpoint pulse with the correct sign and amplitude.

**Friction compensation types**

One of two friction compensation methods can be selected on the SINUMERIK 840D (MD 32490: FRICT_COMP_MODE "Type of friction compensation"):

- Conventional friction compensation (MD 32490: FRICT_COMP_MODE = 1)

With this type, the intensity of the compensation pulse can be set according to the characteristic as a function of the acceleration. This characteristic must be determined and parameterized during start-up using the circularity test. The procedure for this is relatively complicated and requires some experience.

Conventional friction compensation can also be used with SINUMERIK FM-NC.

- Quadrant error compensation with neural networks
  (option on SINUMERIK 840D) (MD 32490: FRICT_COMP_MODE = 2)

To simplify start-up, the compensation characteristic no longer has to be set manually by the start-up engineer but is determined automatically during a training phase and then stored in the non-volatile user memory.

The neural network can reproduce a compensation curve of much better quality and precision.

The function also allows simple automatic re-optimization directly at the machine.

**Circularity test**

The friction compensation function (both conventional and neural friction compensation) can be started up most easily by means of a circularity test. This is done by following a circular contour, measuring the actual position and representing the deviations from the programmed radius (especially at the quadrant transition points) graphically. The measurements are recorded using a “Trace” that is stored in the passive file system.

The circularity test is an “installation tool” function. With the MMC101 or MMC102/103, this function can be selected directly in the Diagnosis area.

See Section 2.7 for more detailed information about the circularity test.

### 2.5.2 Conventional friction compensation

**Type of friction compensation**

Conventional friction compensation is selected by entering the value 1 in MD 32490: FRICT_COMP_MODE (friction compensation type).

**Amplitude adaptation**

In many cases, the injected amplitude of the friction compensation value does not remain constant over the whole acceleration range. For example, for optimum compensation with high accelerations, a smaller compensation value must be injected than for smaller accelerations. For this reason, friction compensation **with adapted injection amplitude** can be activated in cases where high accuracy is required (see figure below). The function is activated axis-specifically in MD 32510:
FRICT_COMP_ADAPT_ ENABLE = 1 (adaptation for friction compensation active).

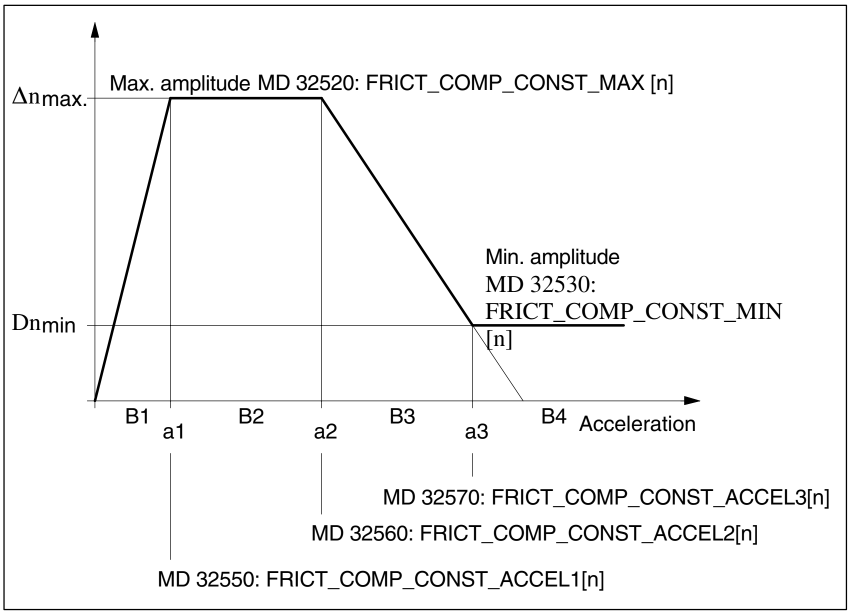



Fig. 2-16 Typical curve for friction compensation with amplitude adaptation

The adaptation characteristic is divided into four ranges (a different injection amplitude $\Delta n$ is applied in each range):

| | | |
|---|---|---|
| B1: | for $a < a_1$ | $\Delta n = \Delta nmax * a / a_1$ |
| B2: | for $a_1 \le a \le a_2$ | $\Delta n = \Delta nmax$ |
| B3: | for $a_2 < a < a_3$ | $\Delta n = \Delta nmax * (1 - (a - a_2) / (a_3 - a_2))$ |
| B4: | for $a \ge a_3$ | $\Delta n = \Delta nmin$ |

**Characteristic parameters**

The parameters of the adaptation characteristic in the figure above must be entered as machine data for specific axes.

| | | |
|---|---|---|
| $\Delta n$ | = | Injection amplitude of the friction compensation value |
| $\Delta n_{max}$ | = | Maximum friction compensation value<br>MD 32520: FRICT_COMP_CONST_MAX [n] |
| $\Delta n_{min}$ | = | Minimum friction compensation value<br>MD 32530: FRICT_COMP_CONST_MIN [n] |
| $a_1$ | = | Adaptation acceleration value 1 for friction compensation<br>MD 32550: FRICT_COMP_ACCEL1 [n] |
| $a_2$ | = | Adaptation acceleration value 2 for friction compensation<br>MD 32560: FRICT_COMP_ACCEL2 [n] |
| $a_3$ | = | Adaptation acceleration value 3 for friction compensation<br>MD 32570: FRICT_COMP_ACCEL3 [n] |

**Note about shape of characteristic**

In special cases, the calculated characteristic may deviate from the typical shape illustrated in the figure above.
In some cases, the value for $\Delta n_{min}$ (MD 32530: FRICT_COMP_CONST_MIN) may even be greater than $\Delta n_{max}$ (MD 32520: FRICT_COMP_CONST_MAX).

## 2.5.3 Start-up of conventional friction compensation

**Circularity test**

The friction compensation function can be started up most easily by means of a circularity test. Here, deviations from the programmed radius (especially at the quadrant transitions) can be measured and displayed while traversing a circular contour.

**Step-by-step start-up**

The conventional friction compensation function must first be selected. (MD 32490: FRICT_COMP_MODE=1).

The friction compensation value mainly depends on the machine configuration. Installation is performed in two stages.

- Stage 1: Calculation of the compensation values without adaptation
- Stage 2: Calculation of the adaptation characteristic (if the friction compensation is dependent on the acceleration and the results of stage 1 are not satisfactory).

## Installation stage 1: Friction compensation without adaptation

**1. Circularity test without friction compensation**

A circularity test without friction compensation
(MD 32500: FRICT_COMP_ENABLE = 0) must be performed first.
The circularity test procedure is described in Section 2.7.

A typical characteristic of quadrant transitions without friction compensation is shown in the figure below.

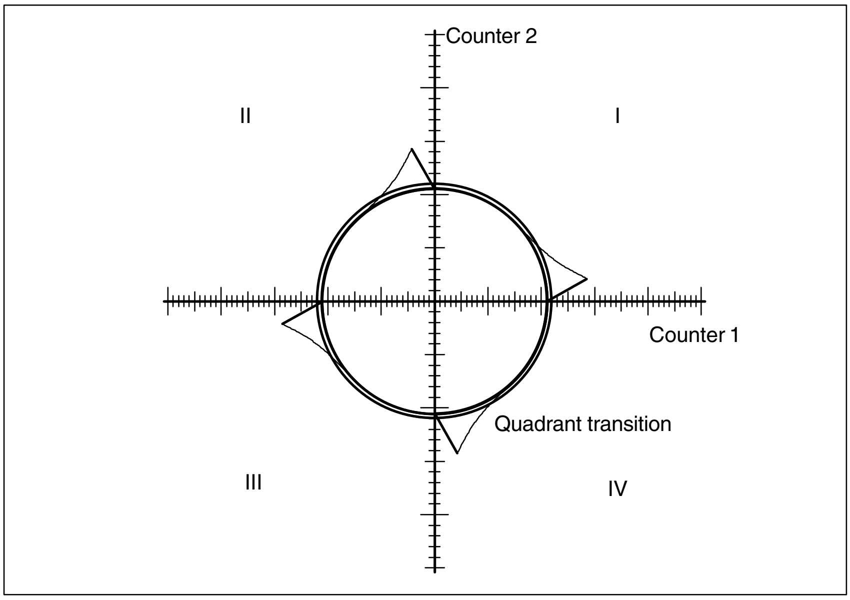



Fig. 2-17 Uncompensated radius deviation at quadrant transitions

**2. Enabling the friction compensation**

After this, the friction compensation must be activated for the axis/spindle in question.

Activate friction compensation
⇒ MD 32500: FRICT_COMP_ENABLE[n] = 1

**3. Deactivate adaptation**

In order to start up friction compensation without adaptation, the adaptation must be deactivated.

Deactivate adaptation
⇒ MD 32510: FRICT_COMP_ADAPT_ENABLE[n] = 0

**4. Determine compensation parameters**

Friction compensation without adaptation is defined by the following parameters:

1. MD 32520: FRICT_COMP_CONST_MAX [n] friction compensation value (amplitude) in [mm/min]
2. MD 32540: FRICT_COMP_CONST_TIME [n] friction compensation time constant in [s]

These two parameters are changed until the circularity test produces minimum or no deviations from the programmed radius at the quadrant transitions (see the next four figures). The tests must be performed with different radii and velocities (typical values for the application of the machine).

**Start value**

A relatively low injection amplitude plus a time constant of a few position controller cycles should be entered as the start value when measuring commences.

**Example:**

MD 32520: FRICT_COMP_CONST_MAX[n] = 10 (mm/min)
MD 32540: FRICT_ COMP_TIME [n] = 0.008 (8msec)

The effect of changing the parameters must be checked using the measured and plotted circles.

**Averaging**

If it is not possible to determine a common compensation time constant for the varying radii and velocities, then the average of the calculated time constants must be worked out.

**Good friction compensation setting**

When the friction compensation function is well set, quadrant transitions are no longer noticeable (see figure below).

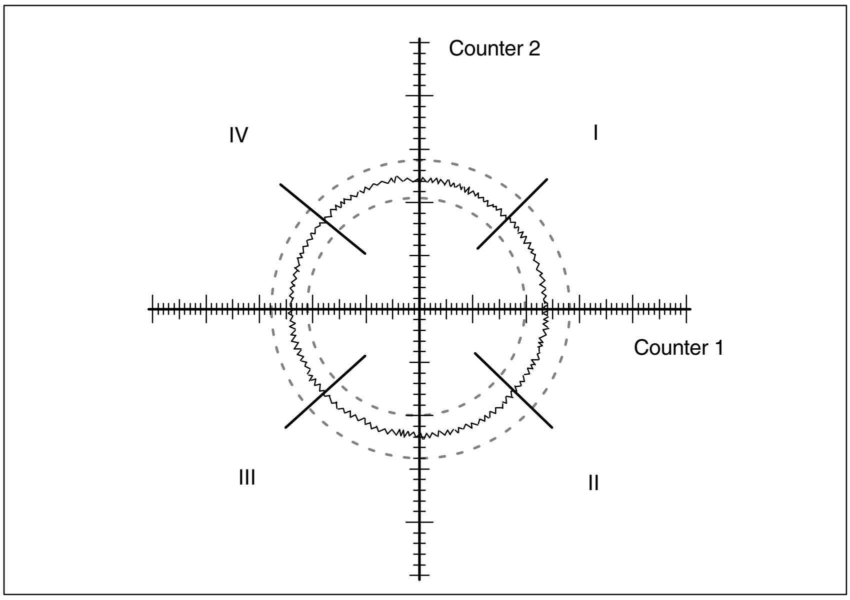



Fig. 2-18 Quadrant transitions with correctly set friction compensation

**Amplitude too low**

When the injection amplitude setting is too low, radius deviations from the programmed radius are compensated inadequately at quadrant transitions during circularity testing (see figure below).

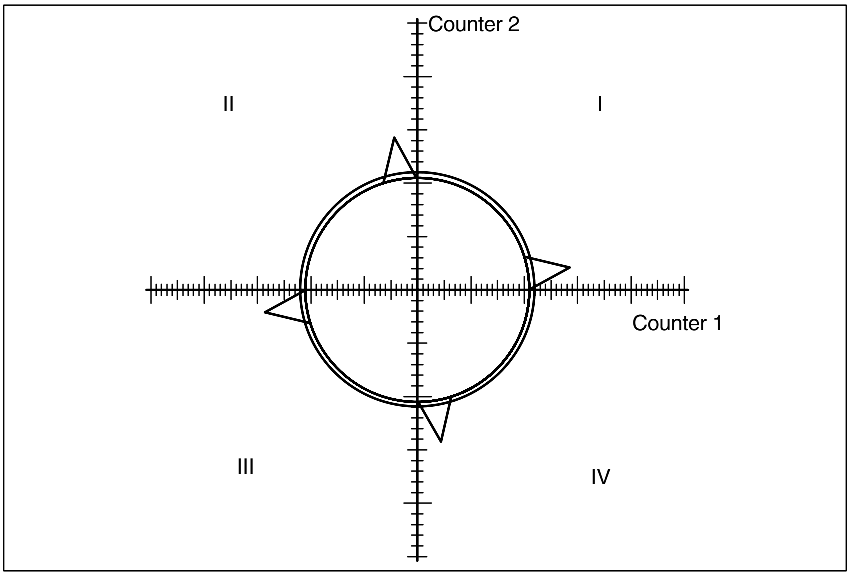



Fig. 2-19 Amplitude too low

**Amplitude too high**

When the injection amplitude setting is too high, radius deviations at quadrant transitions are manifestly overcompensated at quadrant transitions (see figure below).

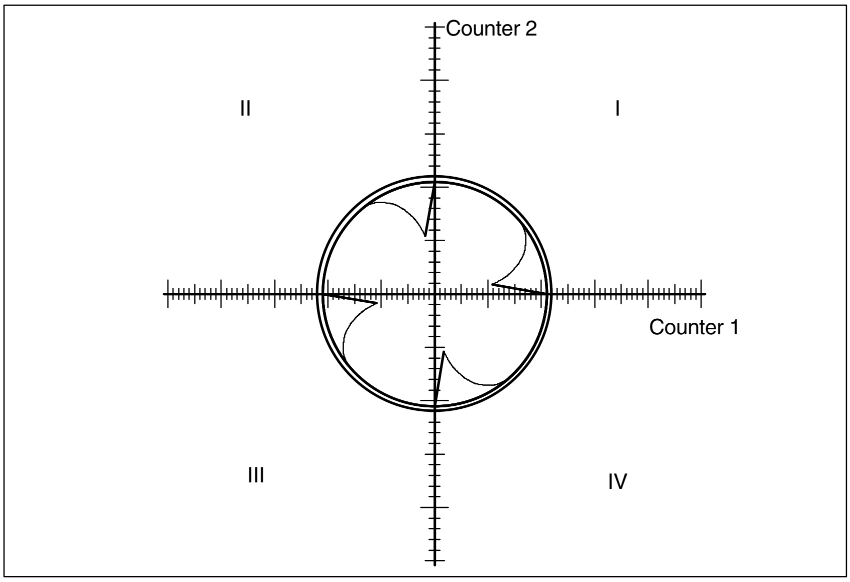



Fig. 2-20 Amplitude too high

**Time constant too low**

When the compensation time constant settings are too low, radius deviations are compensated briefly at quadrant transitions during circularity testing, but are followed immediately again by greater radius deviations from the programmed radius (see figure below).

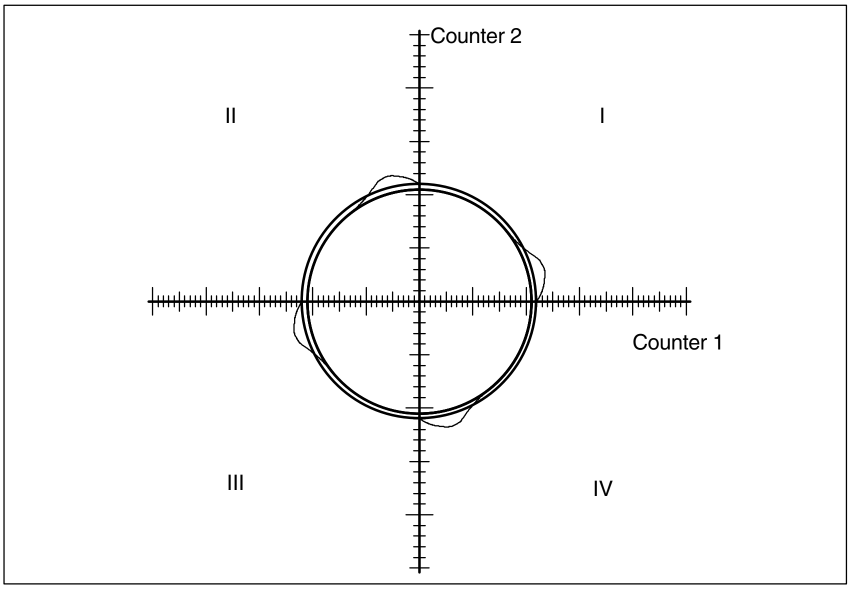



Fig. 2-21 Compensation time constant too small

**Time constant too high**

When the compensation time constant settings are too high, radius deviations are compensated at quadrant transitions during circularity testing (on condition that the optimum injection amplitude has already been calculated), but the deviation in the direction of the arc center increases significantly after quadrant transitions (see figure below).

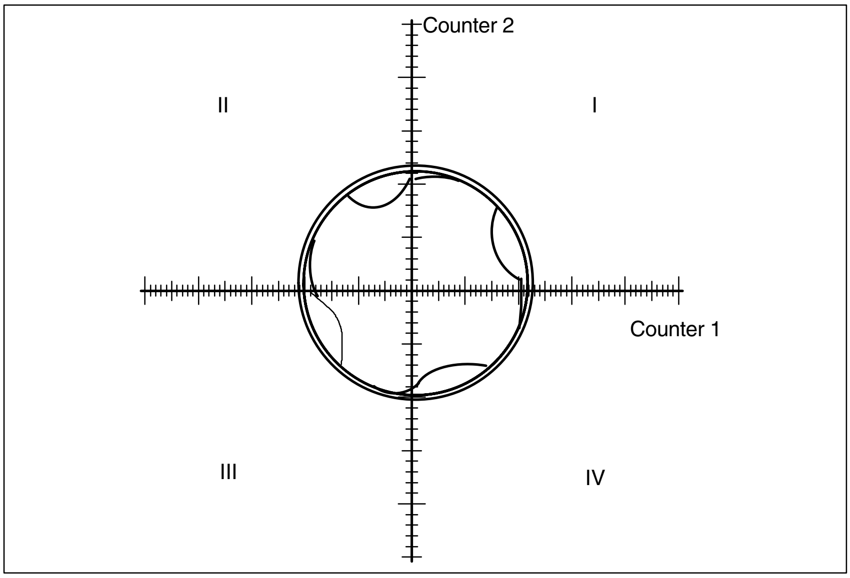



Fig. 2-22 Compensation time constant too large

**Adaptation yes/no?**

If, with the time constant and the constant injection amplitude determined, a good result is achieved both in the circularity test and in positioning over the whole working area (i.e. for all radii and velocities of relevance), curve adaptation will not be necessary.

However, if the friction compensation turns out to be dependent on the acceleration, the adaptation characteristic must be calculated in second stage (see stage 2: Friction compensation with adaptation).

## Installation stage 2: Friction compensation with adaptation

**Application**

Whenever friction compensation depends on the acceleration and the required results cannot be obtained with constant injection amplitude, adaptation must be used.

In order to obtain optimum compensation over the whole of the working range of the friction feedforward control where high demands are made on accuracy, the acceleration dependency of the compensation value must be calculated. To achieve this, the dependency must be measured at various points in the working range between acceleration zero and the maximum planned acceleration. The adaptation characteristic derived from the measurement results is then entered in the above machine data axis-specifically.

**1. Determining the adaptation characteristic**

For different radii and velocities ...

1. ... it is necessary to determine the required injection amplitudes,
2. ... it is necessary to check the compensatory effect of the injection amplitudes using the circularity test,
3. ... it is necessary to log the optimum amplitudes.

The adaptation characteristic (for example, see Fig. 3.8) is defined completely by determining the parameters specified in Subsection 2.5.2. However, many more measured values must be obtained for checking purposes. It must be ensured that there is a sufficiently large number of interpolation points for small radii at high speed. The size of the curves must be obtained by plotting.

**2. Determining acceleration values**

During circular movement, the axial acceleration values are calculated using the radius r and the traversed velocity v according to the formula
$a = v^2/r$ .
Using the feedrate override switch it is easy to vary the velocity and therefore axial acceleration value a.

Acceleration values $a_1$, $a_2$ and $a_3$ for the adaptation characteristic must be entered in MD 32550: FRICT_COMP_ACCEL1 to MD 32570: FRICT_COMP_ACCEL3 in compliance with the condition $a_1 < a_2 < a_3$. If the curve is wrongly parameterized, the alarm 26001 “Parameterization error for friction compensation” is output.

**Example of characteristic settings**

1. Calculation of acceleration
   The axial acceleration during the passage through zero of the speed for a circular path is calculated using the formula $a = v^2/r$.
   With the radius r = 10 mm and a circular velocity of v = 1 m/min (=16.7 mm/s) the acceleration is therefore a = 27.8 mm/s$^2$.
2. Input of curve knee points
   The following accelerations were calculated to be the curve knee points:
   $a_1$ = 1.1 mm/s$^2$ ; $a_2$ = 27. 8 mm/s$^2$ ; $a_3$ = 695 mm/s$^2$

The following values are therefore entered in the machine data in this order:

| | | |
|---|---|---|
| MD 32550: FRICT_COMP_ACCEL1 [n] | = 0.0011 | [m/s$^2$] |
| MD 32560: FRICT_COMP_ACCEL2 [n] | = 0.0278 | [m/s$^2$] |
| MD 32570: FRICT_COMP_ACCEL3 [n] | = 0.695 | [m/s$^2$] |

For example, the following values were calculated for the injection amplitudes:

MD 32520: FRICT_COMP_CONST_MAX [n] = 30 [mm/min]
MD 32530: FRICT_COMP_CONST_MIN [n] = 10 [mm/min]

---

**Note**

If the results obtained at very low velocities are not satisfactory, then the computational resolution for linear positions MD 10200: INT_INCR_ PER_MM or for angular positions MD 10210: INT_INCR_PER_DEG must be increased.

See also MD 32580: FRICT_COMP_INC_FACTOR
(weighting factor of friction compensation value for short traversing movements).

---

# 2.6 Neural quadrant error compensation

## 2.6.1 Fundamentals

**Principle of QEC**

As explained in Section 2.5, the purpose of quadrant error compensation (QEC) is to reduce contour errors occurring during reversal as the result of drift, backlash or torsion. Compensation is effected through prompt injection of an additional speed setpoint (see figure below).

In conventional QEC, the intensity of the compensation pulse can be set according to a characteristic as a function of the acceleration. This characteristic must be calculated and parameterized by a circularity test during start-up (see Fig. 3.8). The procedure for this is relatively complicated and requires some experience.

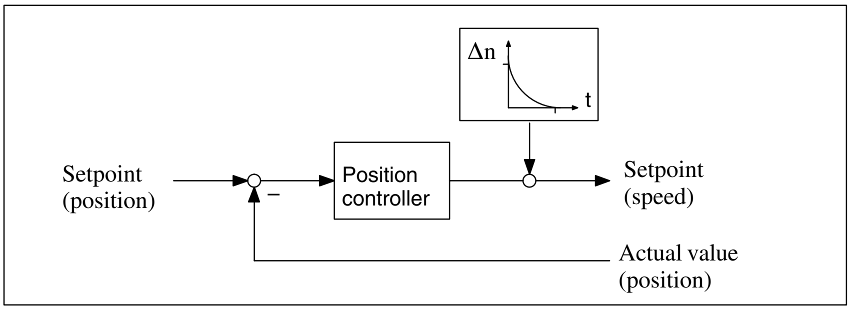



Fig. 2-23 Injection of an additional speed setpoint pulse

**Advantages of QEC with neural network**

On the SINUMERIK 840D with Software Version 2 or higher, the characteristic block that used to be manually parameterized can now be replaced by a neural network. This has the following advantages:

- Start-up has been simplified because the compensation characteristic no longer needs to be set manually by the start-up engineer but is **determined automatically** during a learning phase.
- The characteristic for a manually parameterized friction compensation is approximated by a polygon with 4 straight lines (see Fig. 3.8). For improved precision, the neural network can **reproduce the real curve much better**.

  The resolution of the characteristic curve can be adapted to the precision requirements and a directional quality of the compensation amplitude can be considered.

  In addition to the compensation amplitude, it is possible to adapt the decay time to the acceleration in special cases.
- The system permits **simple automatic re-optimization on site** at any time.

**Requirements for neural QEC**

An essential requirement for implementing QEC with neural network is that the errors occurring on the workpiece at quadrant transitions are detected by the measuring system. This is only possible either with a direct measuring system, with an indirect measuring system with clear reactions of the load on the motor (i.e. rigid mechanics, little backlash) or with suitable compensation. With indirect measuring systems, any backlash that might occur must be compensated by backlash compensation.

**Learning/working phases**

QEC with neural network involves the following two phases:

- **Learning phase**
  During the learning phase, a certain pattern of behavior is memorized in the neural network. The relation between the input and output signals is learnt. The result is the learnt compensation characteristic that is stored in the non-volatile user memory. Activation and deactivation of the learning process is programmed in the NC parts program using special high-level language commands.
- **Working phase**
  During the working phase, additional speed setpoint pulses are injected in accordance with the learnt characteristic. The stored characteristic does not change during this phase.

The learning phase can be executed for several (up to 4) axes at the same time. For further information about training the neural network, see Subsection 2.6.3.

The learning and working phases and the resulting neural QEC are purely axial. There is no mutual influence between the axes.

**Saving characteristic values**

On completion of the learning phase, the calculated compensation data (characteristic values in user memory) including the network parameters (QEC system variables) must be saved in a file selected by the operator. As a standard, these files are called “AXn_QEC.INI”.

**Loading characteristic values**

These saved and learned compensation data can be loaded back directly to the user memory in the same way as part programs.

When the parts program containing the tables is loaded, the compensation values are transferred to the NC user memory.
The characteristic values become effective only after compensation has been enabled.

Characteristic values **cannot** be written when the compensation function **is active** (MD 32500: FRICT_COMP_ENABLE must be set to 0 and activated).

With QEC: The QEC must be enabled (and activated) by setting MD 32500: FRICT_COMP_ENABLE = 1 (QEC active).

**Recommended default setting for start-up**

As mentioned above, the neural network integrated in the control automatically adapts the optimum compensation data during the learning phase.

The axis involved must perform reversals with acceleration values constant section by section. Before activation of the learning phase, the parameters of the neural network (QEC system variables) must be pre-assigned in accordance with the requirements.

In order to simplify start-up as much as possible, NC programs are provided as reference examples.

As described in Subsection 2.6.4, the start-up engineer must first learn the characteristic for the axes using these reference examples and the recommended QEC parameter values and check the contour accuracy achieved using the circularity test (see Section 2.7). If the results do not meet the requirements, re-optimization must be performed changing the parameters appropriately (see Subsections 2.6.2, 2.6.3 and 2.6.5) (i.e. relearning).

## 2.6.2 Parameterization of neural QEC

**Machine Data**

The basic configuring data for the neural QEC are stored as machine data.

- MD 32490: FRICT_COMP_MODE
  Method of friction compensation (2 = neural QEC)
- MD 32500. FRICT_COMP_ENABLE
  Friction compensation active
- MD 32580: FRICT_COMP_INC_FACTOR
  Weighting factor for friction compensation value for short traversing blocks
- MD 38010: MM_QEC_MAX_POINTS
  Maximum number of compensation values for QEC with neural networks

With these machine data, the neural QEC is activated as soon as the memory space is reserved in the non-volatile RAM. The procedure and assignment is described in Subsection 2.6.4 “Start-up” or in Chapter 4.

All other data are set using system variables.

**QEC system variables**

The data for parameterizing the neural network are defined as system variables that can be written and read by an NC program. The following system variables are used for parameterization of the neural network:

- **$AA_QEC_COARSE_STEPS** “Coarse quantization of the characteristic”

  This parameter defined the coarse quantization of the input signal and is therefore the resolution of the characteristic. The larger the value that is selected, the higher the memory requirement and the greater the length of time required for the training phase. See the end of this section for more information.

  Value range: 1 to 1024; recommended value: 49

- **$AA_QEC_FINE_STEPS** “Fine quantization of the characteristic”

  This parameter defines the fine quantization of the input signal and is therefore the resolution of the characteristic. The larger the value that is selected, the higher the memory requirement.

  Value range: 1 ... 16; recommended value: 8

- **$AA_QEC_DIRECTIONAL** "Directionality"

  This parameter defines whether the compensation is to be injected directionally or not. If activated, a separate characteristic is determined and stored for each acceleration direction. Because two characteristics are used, double the memory space must be reserved in the non-volatile user memory.

  Value range: TRUE/FALSE; recommended value: FALSE

- **$AA_QEC_LEARNING_RATE** "Learning rate for the active learning"

  With the learning rate it is possible to determine how quickly the optimum characteristic is to be learnt in the active learning phase of the neural QEC. This value is a weighting factor with which it is possible to define to what extent the deviations affect the injection amplitude. With higher values (>100%), the characteristic is learned more quickly but too high learning rate values (weighting factors) can cause instability (two-step response).

  A small learning rate is recommended for relearning processes during normal operation (<50%) otherwise the characteristic is changed on every little disturbance when the speed passes through zero.

  Value range: > 0%; $\leq$ 500%; recommended value: 50%

- **$AA_QEC_ACCEL_1 / _2 / _3** "Acceleration limit values for the characteristic areas 1 / 2 / 3"

  The acceleration characteristic is divided into three areas. In each area there is a different quantization of the acceleration steps. In the low acceleration range, an especially high resolution is required for the characteristic in order to reproduce the widely varying compensation values there. For this reason, the input signals are quantized more finely, the smaller the acceleration is.

  Recommended values for
  - $AA_QEC_ACCEL_1 20 mm/sec$^2$ (= 2% of $AA_QEC_ACCEL_3)
  - $AA_QEC_ACCEL_2 600 mm/s$^2$ (= 60% of $AA_QEC_ACCEL_3)
  - $AA_QEC_ACCEL_3 1000 mm/s$^2$ (maximum acceleration of working range)

  The value of the parameter $AA_QEC_ACCEL_3 must be entered as appropriate to the requirements; i.e. the neural network only works and learns optimally in this range. If a higher acceleration is detected than the parameterized working area, the injection amplitude that was determined during the defined maximum acceleration of the working range is used. At high accelerations, this injection value is relatively constant.

  The recommended values must only be changed if the compensation is insufficient in these acceleration ranges. For further information see Subsection 2.6.5.

- **$AA_QEC_TIME_1** "Time constant for the neural QEC (decay time)"

  With this, the decay time of the compensation setpoint pulse is set if adaptation of the decay time is not used.

The optimum decay time must be ascertained manually using the circularity test at a working point in the mid acceleration range. The procedure is detailed in the section dealing with friction compensation (Subsection 2.5.3) (analogous to MD 32540: FRICT_COMP_TIME).

With the recommended value (15 ms), it is possible to achieve good results.

Value range: $\geq$ 0; recommended value: 0.015s

If the decay time adaptation is active, then $AA_QEC_TIME_1 determines the filter time constant in the center of the operating range (i.e. with 0.5 * $AA_QEC_ACCEL_3).

- **$AA_QEC_TIME_2** "Compensation time constant for adaptation of the decay time of the correction value"

  At a value of zero of less than or equal to $AA_QEC_TIME_1, no adaptation is performed.

  The decay time is usually constant over the entire working range. In rare cases however, it can be advantageous to raise the decay time in the very small acceleration range, or to lower it at high accelerations. For further information see Subsection 2.6.5.

  Value range: $\geq$ 0; recommended value: 0.015s (identical to $AA_QEC_TIME_1)

- **$AA_QEC_MEAS_TIME_1 / _2 / _3** "Measurement duration for determining the error criterion in the acceleration range 1 / 2 / 3"

  The measuring time is started, as soon as the criterion for injection of the compensation value is fulfilled (i.e. the set speed changes sign). The end of the measuring item is defined by the set parameter values.

  Different measuring times are required for each characteristic range.

  Recommended values for

  – $AA_QEC_MEAS_TIME_1: 0.090s (= 6 *$AA_QEC_TIME_1)

  – $AA_QEC_MEAS_TIME_2: 0.045s (= 3 * $AA_QEC_TIME_1)

  – $AA_QEC_MEAS_TIME_3: 0.030s (= 2 * $AA_QEC_TIME_1)

  The recommended values must only be changed if the compensation is insufficient in these acceleration ranges or if $AA_QEC_TIME_1 is changed. For further information see Subsection 2.6.5.

**Transfer of parameters**

The QEC system variables are stored in the non-volatile user memory after the NC program is started where they remain unchanged until the memory is erased or reformatted or until a new learning or relearning process takes place or until they are written by the NC program.

Before the learning cycle is called, all system variables must be assigned valid values for the learning process. For example, this can be done in a subprogram. After this NC program has run and a reset has been performed, the QEC data are active.

**Characteristic data**

The characteristic data determined during the learning process are stored as system variables in the user memory reserved for this purpose.

Format: **$AA_QEC**[**n**] Range of n: 0 to 1024

These values write the learned characteristic in internal formats and must therefore **not be changed**!

**Quantization of characteristic**

The quantization, and thus the resolution, of the characteristic is defined via the two quantities **fine quantization** ($AA_QEC_FINE_STEPS) and **coarse quantization** ($AA_QEC_COARSE_STEPS). The finer the resolution, the higher the memory requirement and the longer the duration of time required for the learning phase.

The number of memory locations required and the total number of quantization intervals is calculated by the formula:

Number of memory locations = $AA_QEC_FINE_STEPS * ($AA_QEC_COARSE_STEPS+1)

Up to 1025 memory locations per axis can be reserved. In this way, a sufficiently high resolution is achieved for high precision requirements.

The following 3 figures illustrate the meaning of the characteristic values for coarse and fine quantization, and their effect on the teach-in period, as a function of the parameter “Detailed learning active y/n”. Three cases are distinguished for better understanding.

Case 1: Coarse quantization > 1; fine quantization = 1
(special case; usually the fine quantization is in the region of eight):

In this case, the interpolation points of the characteristic are determined solely by coarse quantization (see figure below).

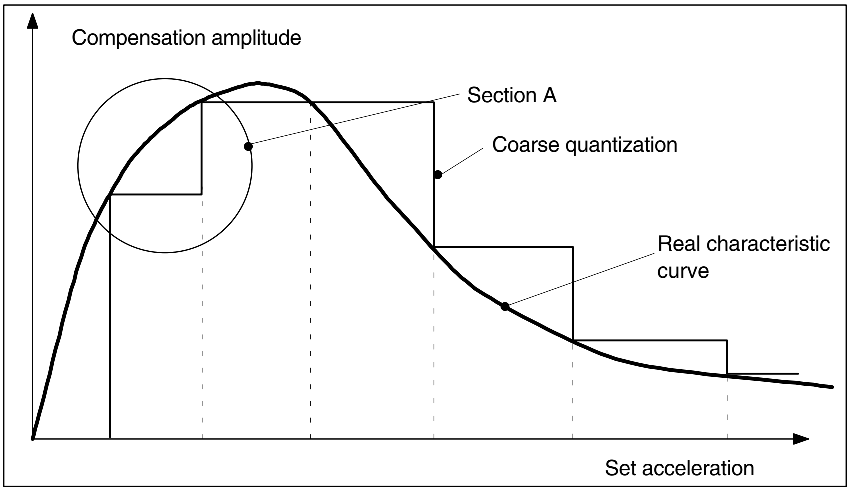



Fig. 2-24 Coarse quantization of characteristic

Case 2: Coarse quantization > 1; fine quantization > 1; “Detailed learning” is deactivated (this setting is the default);

In this case, discrete linear interpolation is used for fine quantization between the interpolation points of the coarse quantization.

The learning duration is identical with 1 because learning only occurs at the interpolation points of the coarse quantization.

The effect of fine quantization on a section of characteristic within a coarse quantization process is shown in the figure below (see also Section A in figure above).

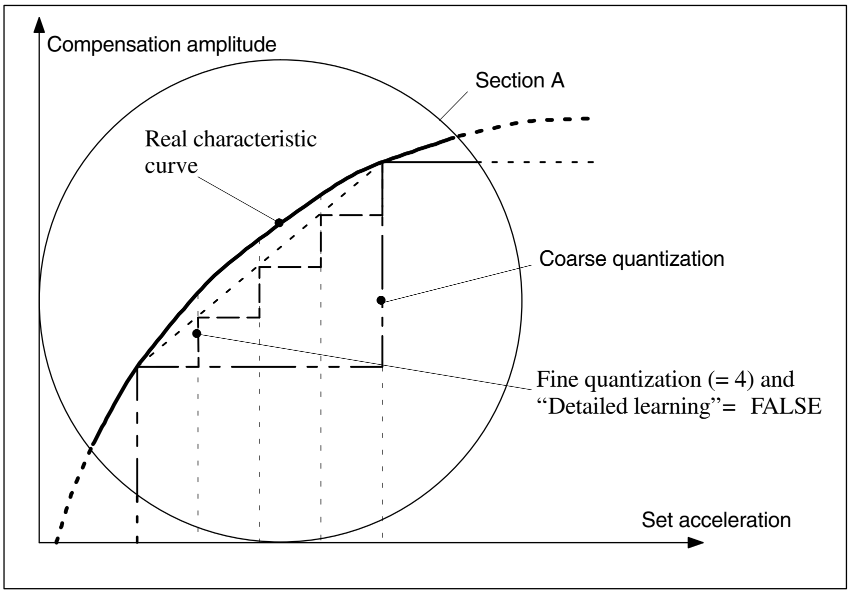



Fig. 2-25 Effect of fine quantization with “Detailed learning” inactive

Case 3: Coarse quantization > 1; fine quantization > 1; “Detailed learning” is activated (its use is only recommended for very high precision requirements):

With “Detailed learning”, learning occurs both at the interpolation points of the coarse quantization and of the fine quantization.

The learning duration is therefore much longer.

The figure below shows a severely fluctuating characteristic curve on which the effect of selecting and deselecting the “Detailed learning” function is clear.

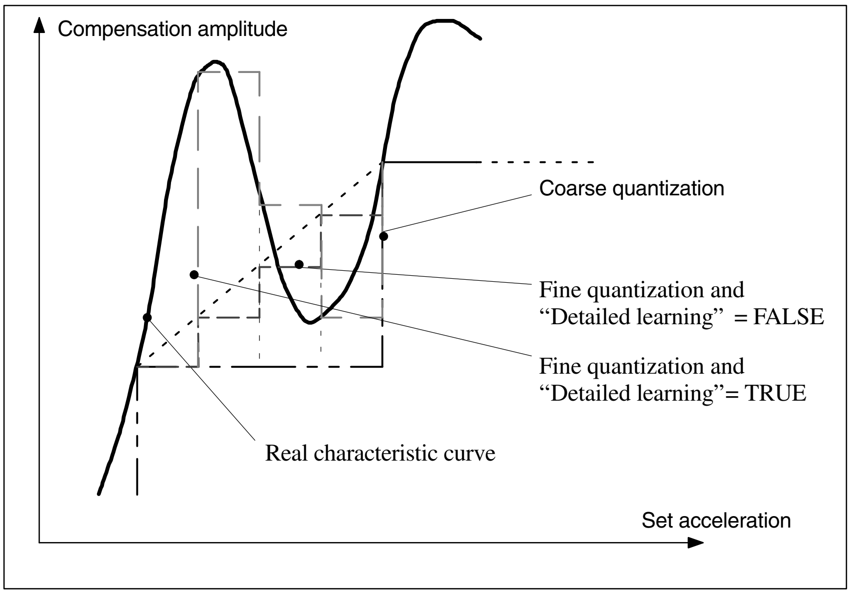



Fig. 2-26 Effect of fine quantization with “Detailed learning” active

## 2.6.3 Learning the neural network

**Learning process sequence**

A certain type of response is impressed upon the neural network during the learning phase. The relation between the input and output signals is learned.

The learning process is controlled entirely by NC programs and is divided into the following areas:

1. Preset the QEC system variables for the learning process
2. Activate QEC system variables (by starting the NC program)
3. Parameterize the learning cycle
4. Start the learning cycle

The result is the learnt compensation characteristic that is stored in the non-volatile user memory.

The results achieved must be checked using the circularity test (Section 2.7).

**Reference NC programs**

In order to ease the task of the start-up engineer in starting up the QEC with neural networks, NC programs containing specimen routines for learning movements and assignments of QEC system variables (recommended values) are available.

These are the following reference NC programs:

- QECLRNP.SPF — Learning with POLY standards (Option “POLY” necessary)
- QECLRNC.SPF — Learning with circles
- QECDAT.MPF — Reference NC program for assigning system variables and for parameterization the learning cycle
- QECSTART.MPF — Reference NC program that calls the learning cycle

These NC programs are contained on the diskette of the basic PLC program for the SINUMERIK 840D.

Implementing the learning process solely via NC programs has the following advantages:

- Learning can be fully automatic without operator intervention.
  This is advantageous for series start-ups if the optimum learning parameters for a machine type have been found and only the characteristic for each individual machine remains to determined or retrained .
- Learning can be executed for several axes (up to 4) simultaneously.
  This reduces the learning phase for the machine considerably.
- The traverse movements can easily be adapted to special requirements.
- Start-up of the neural QEC is possible even where a simple MMC is used (e.g. MMC 100) (exception: a circularity test on the MMC is only possible with MMC 101-103; otherwise use an installation tool).

**Learning motion**

The axis traversing motions that must be executed to learn a specific response are generated by an NC program. Each learning motion of the sample learning cycle comprises a group of NC blocks with parabolic movements (ensuring that the axis traverses at the most constant possible setpoint speed after the zero crossing; see figure below) in which the axes oscillate at constant acceleration in each program section. The acceleration is decreased from group to group. In the figure below, NC blocks 2 to 3, 5 to 6 and 8 to 9 each form a group; the transitional movements to lower acceleration rates are programmed in blocks 1, 4, 7 and 10.

**Note**

So that the learning parameters act as preset, the feedrate override switch must be set to 100% during the learning phase.

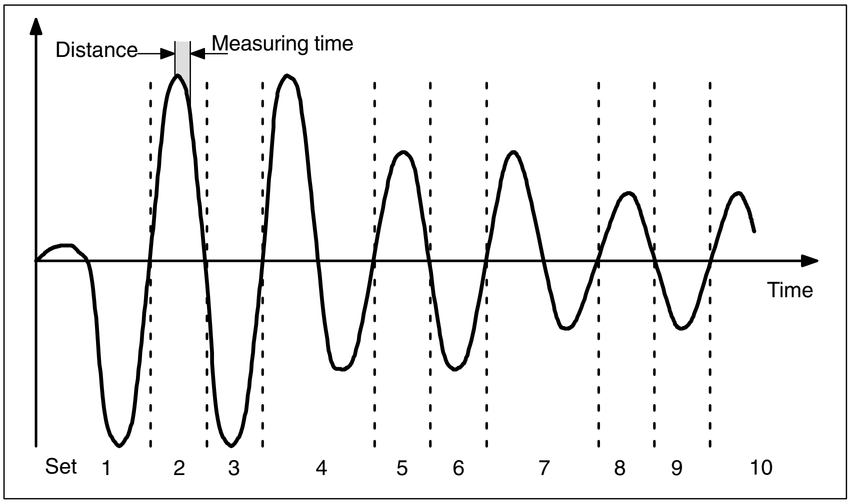



Fig. 2-27 Typical traverse motion of an axis when learning the QEC characteristic

**Assignment of system variables**

Before a learning cycle is called, all QEC system variables must be set to the values required for the learning process. The values recommended in the reference NC program must be checked and changed if necessary (see Subsection 2.6.2).

**Learning ON / OFF**

The actual learning process of the neural network is then activated in the reference NC program. This is done using the following high-level language command:

**QECLRNON(axis name 1, ... 4)** Learning ON (for specified axes)

Only during this phase are the characteristics changed.

After the learning motions of the required axes have been completed, the learning process is deactivated for all axes. This is done with the high-level language command:

**QECLRNOF** Deactivate learning (simultaneously for all axes)

After power-on reset, end of program (M02/M30) or operator panel front reset, learning is also deactivated.

The current “Learning on / off” status is displayed in the service display “Axes” with “QEC learning active” (1 = active; 0 = inactive).

**Learning cycle call**

Once learning has been activated, the reference NC program calls the learning cycle by means of the following input parameters:

- **Number of axes** to which learning is to apply (up to four).

  Prerequisite: If more than one axis is to learn at the same time, all QEC system variables of the axes involved must have the same values. These values are monitored and an error message is output if they are not equal.

- **Names of the learning axes**
- **Initial number** (same for all axes)
  Value always 0 (setpoint branch)
- **Learning mode** (initial learning = 0; relearning = 1)

  0: Initial learning active. All values of the network are preset to 0 before learning.

  1: Relearning active. Learning continues with the values already learnt in the defined step width.

- **Detailed learning active** yes/no (TRUE/FALSE)

  FALSE: “Detailed learning” is not active. The characteristic is therefore learnt in the step width of the coarse quantization of the acceleration.

  TRUE: “Detailed learning” is active. The characteristic is therefore learnt in the step width of the fine quantization of the acceleration, i.e. with fine quantization of 10 steps per coarse step, determination of the characteristic takes ten times longer. This parameter must therefore only be used for extremely high precision requirements.

---

**Note**

If “Detailed learning” is selected, the number of learning passes can and must be reduced in order to reduce the learning duration (recommended range: between 1 and 5).

---

- **Number of learning passes**
  Default value = 15; range > 0

  The effect of this parameter depends on whether "Detailed learning active" is set or not.

  a) Detailed learning not active (= FALSE):
  The number of test motions (back and forth) is defined for each acceleration stage. The higher the number, the more accurate learning is, but the longer learning takes.

  With directional compensation ($AA_QEC_DIRECTION = TRUE), the parameterized number of test movements for every direction is generated.

  b) Detailed learning active (= TRUE):
  In this way, the number of complete passes from maximum to minimum acceleration and vice versa is activated with the fine step width.
  In other words, with a value of 1, all acceleration steps are passed through starting with the maximum value. For every acceleration stage, two test movements are generated if there is no directional compensation ($AA_QEC_DIRECTION = FALSE), otherwise four test movements are performed per acceleration stage.

  A reduction of the "Number of learning passes" can be made if data blocks for the machine type already exist (series machines) and these are to be used as a basis for further optimization.

- **Section-by-section learning active** yes/no (TRUE/FALSE)
  "Section-by-section learning" in certain acceleration ranges is especially interesting for "Detailed learning" e.g. in technologically important ranges of the machine. By defining the ranges appropriatelyit is possible to reduce the learning duration.

  Default = FALSE

- **Range boundaries for "Section-by-section learning"** (minimum acceleration, maximum acceleration); only relevant for "Section-by-section learning active".

  Default value = 0; format: $mm/s^2$

- **Time taken for one test motion** (to and fro)

  Default value = 0.5; format: s (seconds)
  (corresponds to a frequency of 2Hz)

**Requirements**

In the learning phase, the neural QEC requires a speed feedforward control (MD 32620: FFW_MODE=1; FFWON), but no jerk limitation (BRISK). The feedforward control must therefore be correctly parameterized and optimized. When the learning process is started a check is made to see whether the speed feedforward control is activated. If not, the learning process is canceled and an error message is generated.

## 2.6.4 Start-up of neural QEC

**General**

This describes start-up of QEC with neural networks. As we have already mentioned, the compensation characteristics during the learning phase are determined automatically.

The axis involved must perform reversals with acceleration values constant section by section. The QEC system variables for parameterization of the neural network must also be preset to meet the requirements.

To simplify start-up as much as possible, NC programs are provided to serve as reference examples (see Subsection 2.6.3).

In the learning process, a distinction is made between “initial learning” (especially for first start-up) and “relearning” (especially for re-optimization of characteristics already learnt). The procedures of “initial learning” and “relearning” are described below.

If the compensation characteristics for the machine are to be learnt for the first time, we recommend use of the reference NC programs specified in Subsection 2.6.3.

**“Initial learning” sequence**

“Initial learning” –> cycle parameters “Learning mode” = 0

1. a) Activate QEC with neural networks for the required axes:

   MD 32490: FRICT_COMP_MODE = 2
   Note: QEC with neural networks is an option!

   b) Reserve memory space for the compensation points

   MD 38010: MM_QEC_MAX_POINTS
   If the required number is not yet known, a generous amount of memory must be reserved initially (see also item 12).

   c) Parameterize and optimize the speed feedforward control (required for the learning phase)

   d) Perform a hardware reset (because of the re-allocation of the non-volatile user memory).

2. Activate the QEC system variables:
   Adapt the reference NC program QECDAT.MPF for assigning the QEC system variables for all axes concerned (if necessary use the recommended values) and start the NC program. If error messages are output, correct the values and restart the NC program.

3. Create the NC program that moves the machine axes to the required positions and parameterizes and calls the reference learning cycle QECLRN.SPF (as in the example program QECSTART.MPF). The feedrate override switch must be set to 100% of the learning phase so that the parameters can take effect in accordance with the defaults.

4. Activate the learning phase by starting this NC program.
   The compensation characteristic is learnt for all the parameterized axes simultaneously. The learning duration depends on the specified learning parameters. If default values are used, it can take several minutes.
   The status of the axes concerned can be observed in the service display “axis” in the display “QEC learning active”.

5. Activation of the injection of the compensation values for the required axes: MD 32500: FRICT_COMP_ENABLE = 1.
6. Parameterize the trace for the circularity test in the menu “Circularity test measurement” (with MMC101-103 or installation tool).
   Parameter values for the reference NC program:
   Radius[mm]:
   Feedrate[mm/min].
   After this, enable the measuring function with the vertical softkey “start”.
7. Start the NC program with the test motion (circle).
   The position actual values are recorded during the circular movement and stored in the passive file system. After termination of data recording, the recorded contour is displayed as a diagram.
8. Check the quadrant transitions for the contour recorded.
9. Depending on the result, repeat items 2, 4, 7, 8 and 9 if necessary. It might be necessary to change certain QEC system variables first (see also Subsection 2.6.3).
10. The compensation characteristics must be saved as soon as the contour precision meets the requirements (see Subsection 2.6.3).
11. If necessary, the memory area previously reserved for the compensation values can be reduced to the memory actually required.

    **Caution**: When the setting in MD 38010: MM_QEC_MAX_POINTS is altered, the non-volatile user memory is automatically re-allocated on system power-on. All the user data in the non-volatile user memory are lost. These data must therefore be backed up first. After power-on of the control, the backed up characteristics must be loaded again.

**“Relearning” sequence**

“Relearning” –> cycle parameter “Learning mode” = 1
The “Relearning” function allows characteristics that have already been learned to be re-optimized in a simple, automatic process. The values already in the user memory are taken as the basis.

The reference NC programs adapted to the machine (e.g. from “initial learning”) must be used in the learning phase for “relearning”. Generally, the previous values of the QEC system variables can still be used. Before the learning cycle is called, the parameter “learning mode” must be set to 1 (meaning “relearning”). It might also be used to reduce the “number of training passes”.

**Sequence of operations for "Relearning"**

The sequence of operations involved in the Relearning process is described below.

1. If characteristic values have not yet been stored in the user memory (RAM) (e.g. start-up of a series machine), the pre-optimized data block must be loaded (see Subsection 2.6.1).
2. Adapt the NC program that moves the machine axes to the required positions and parameterizes and calls the learning cycle. The parameters for the learning cycle (e.g. QECLRN.SPF) might have to be changed for "relearning".
   – Set "Learn mode" = 1
   – Reduce the "number of learning passes" if necessary (e.g. to 5)
   – Activate "section-by-section learning" if necessary and define the associated range boundaries
3. Activate the learning phase by starting this NC program.
   The compensation characteristic is learnt for all the parameterized axes simultaneously.
4. Parameterize the trace for the circularity test in the menu "Circularity test measurement" (with MMC101-103 or installation tool). After this, enable the measuring function with the vertical softkey "start".
5. Start the NC program with the test motion (circle).
   The position actual values are recorded during the circular movement and stored in the passive file system. After termination of data recording, the recorded contour is displayed on the MMC.
6. Check the quadrant transitions for the contour recorded.
7. Depending on the result, repeat items 3, 4, 5 and 6 if necessary. It might be necessary to change certain QEC system variables first (see also Subsection 2.6.5).
8. The compensation characteristics must be saved as soon as the contour precision meets requirements (see Subsection 2.6.1).

### 2.6.5 Further optimization and intervention options

**Optimization options**

In cases where the results of the circularity test do not meet the required accuracy standards, the system can be further improved by selective changes to QEC system variables. Several ways of optimizing the neural QEC are explained here.

**Alteration of coarse and fine quantization**

As described in previous sections, input variables are quantized by means of the “Coarse quantization” and “Fine quantization” values.

A high value for the fine quantization causes a “similar” output signal to be obtained for adjacent intervals of the input signal, allowing, for example, measuring errors which occur only at a particular acceleration rate to be identified (see Fig. 3.16).

With a low fine quantization, highly fluctuating characteristics are reproduced better.

For the neural friction compensation, it is necessary to make use of the largest error tolerance by setting a high fine quantization ($AA_QEC_FINE_STEPS in the region of 5 to 10).

**Direction-dependent compensation**

Direction-dependent friction compensation must be used in cases where compensation is not applied equally on opposing quadrants when compensation values are being injected independently of direction (see figure below).

The directional injection is activated via the system variable $AA_QEC_DIRECTIONAL = TRUE.

Here, the following aspects must be observed:

- Since a characteristic is learned and stored for every direction of acceleration, double the memory space is required in the non-volatile user memory. MD 38010: MM_QEC_MAX_POINTS accordingly.
- The number of learning passes must be raised because only every second passage occurs at the same location.
- If the characteristic resolution is the same, start-up takes longer.

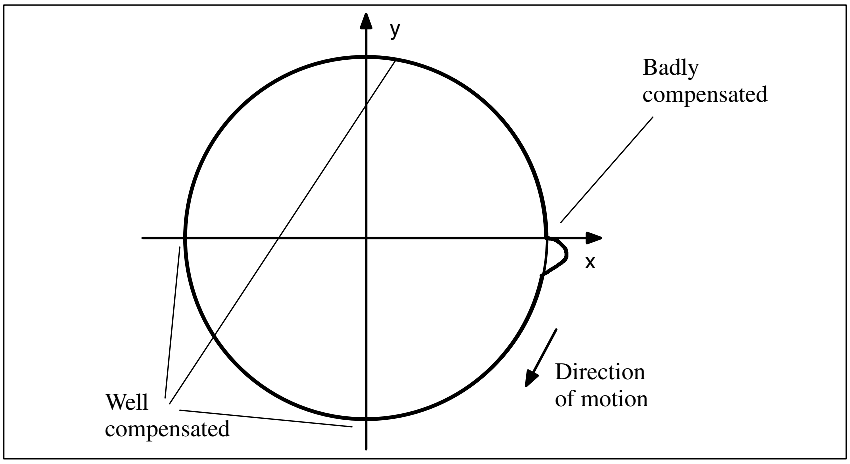



Fig. 2-28 Example of directional friction compensation (circularity test)

**Modification of characteristic ranges**

The acceleration characteristic is subdivided into three ranges. In the low acceleration range, an especially high resolution is required for the characteristic in order to reproduce the widely varying compensation values there. Therefore, the lower the acceleration rate, the finer the quantization of the input quantity (see figure below).

In the high acceleration range, there are only small changes in the compensation values so that a small resolution is perfectly sufficient.

The percentage settings recommended in Subsection 2.6.2 for $AA_QEC_ACCEL_1 (2% of $AA_QEC_ACCEL_3) and for $AA_QEC_ACCEL_2 (60% of $AA_QEC_ACCEL_3) are based on empirical values measured on machines with a maximum acceleration rate (= operating range) of up to approx. 1 m/s$^2$.

If the working range is significantly reduced, then the limit values for $a_1$ and $a_2$ must be set somewhat higher as a percentage of $a_3$. However, $AA_QEC_ACCEL_1 must not exceed the range of approx. 5% of the maximum acceleration. Useful boundaries for $AA_QEC_ACCEL_2 are approx. the values 40% to 75% of the maximum acceleration.

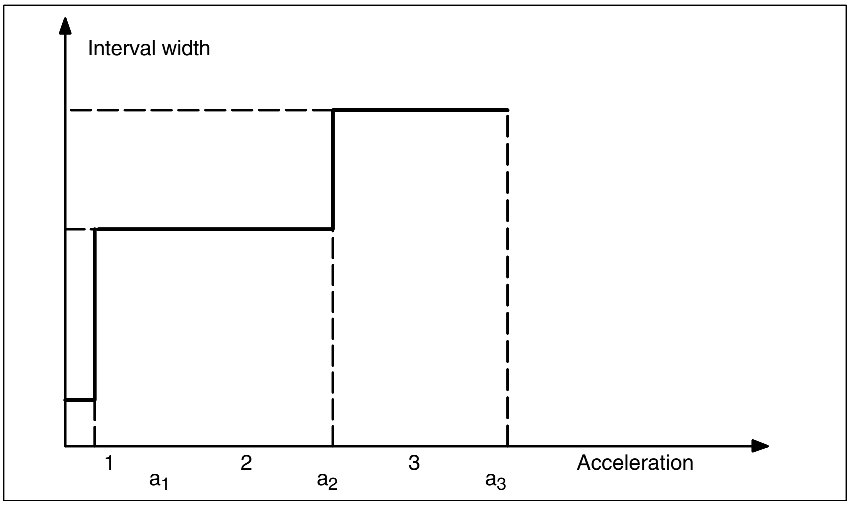



Fig. 2-29 Interval width in acceleration ranges

**Adaptation of the decay time**

In special cases, it is possible to adapt the decay time of the compensation setpoint pulse in addition to the compensation amplitude.

If, for example, the circularity test reveals that in the low acceleration range ($a_1$) the quadrant transitions yield good compensation results but that radius deviations occur again immediately after this, it is possible to achieve an improvement by adapting the decay time.

The time constant without adaptation ($AA_QEC_TIME_1) is only valid in the mid acceleration range (50%).

The adaptation of the decay time for the compensation setpoint impulse according to the characteristic shown in the figure below is parameterized with system variable $AA_QEC_TIME_2 (for acceleration = 0). The adaptation is formed by these two points according to an $e^{-x}$ function (see figure below).

The adaptation is performed under the following condition:
$AA_QEC_TIME_2 > $AA_QEC_TIME_1

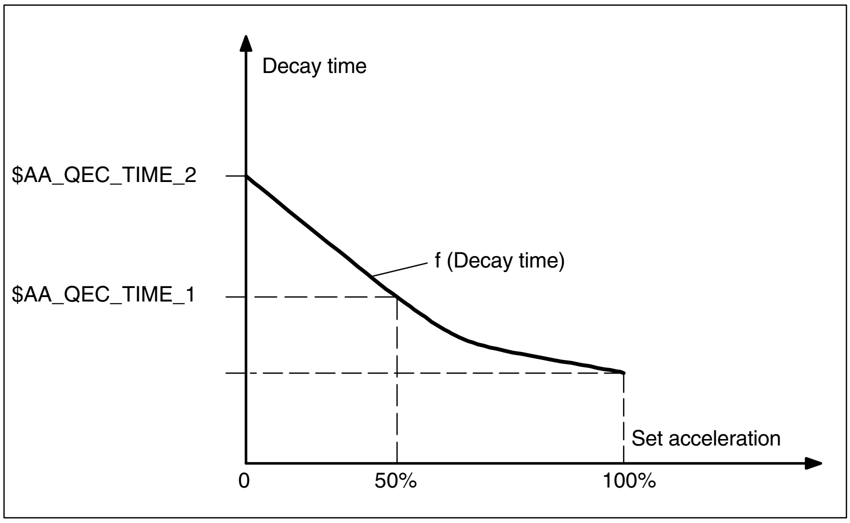



Fig. 2-30 Adaptation of the decay time

**Alteration of error measuring time**

During the learning phase for the neural network, the error measuring time determines the time window within which contour errors are monitored after a zero-speed passage.

Experience has shown that the error measuring time to be used for average acceleration rates (approx. 2 to 50 mm/s$^2$) corresponds to three times the value of the decay time ($AA_QEC_MEAS_TIME_2 = 3 * $AA_QEC_ TIME_1).

In the very low and high acceleration ranges, the error measuring time must be adapted. This is done automatically according to the characteristic in the figure below. The error measurement duration for small accelerations is set to six times the value of the decay time ($AA_QEC_MEAS_TIME_1 = 6 * $AA_QEC_ TIME_1); double the decay time ($AA_QEC_MEAS_TIME_3 = 2 * $AA_QEC_ TIME_1) is taken as the error measurement time for larger accelerations.

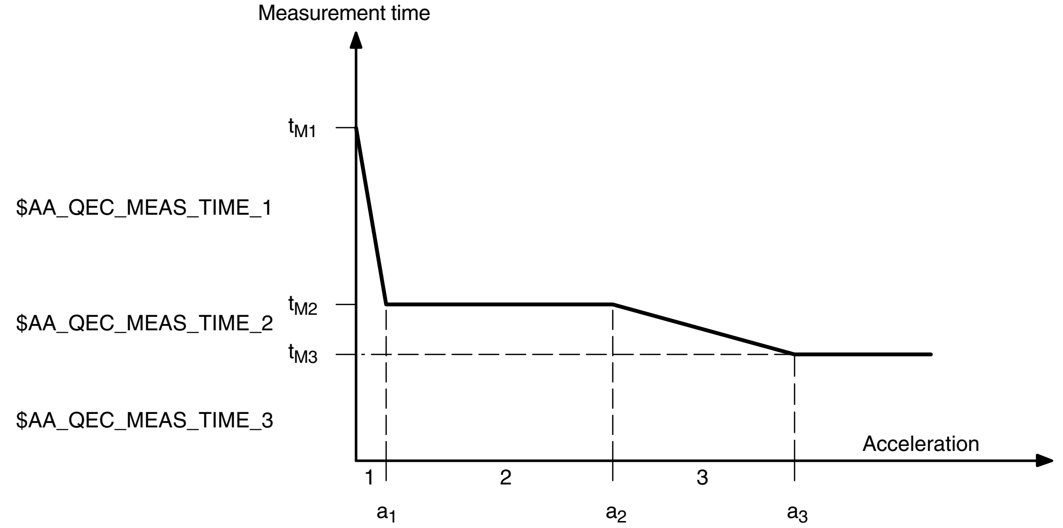



Fig. 2-31 Dependency of error measuring time on acceleration rate

In special cases, it might be necessary to reparameterize the error measuring times:

- Setting very extreme values for the compensation time constant of the QEC. Experience has shown that error measuring times of less than 10 msec and greater than 200 msec are not useful.
- Parameterization of the error measuring times with adaptation of the decay time of the compensation value.

  If the adaptation of the decay time of the compensation value is active (see above), the following rule of thumb is applicable to the parameterization of the error measuring time for acceleration range 1:

  $AA_QEC_MEAS_TIME_1 = 3 * $AA_QEC_ TIME_2

  Example:
  Decay time ($AA_QEC_TIME_1) = 10 ms
  Adaptation of the decay time ($AA_QEC_TIME_2) = 30 ms
  Following the rule of thumb given above,
  the error measuring time for acceleration range 1 is therefore:
  $AA_QEC_MEAS_TIME_1 = 3 * 30 ms = 90 ms
  Without adaptation of the decay time, it would only be
  $AA_QEC_MEAS_TIME_1 = 6 * 10 ms = 60 ms.

**Overcompensation with short traversing motions**

Practical experience has shown that the optimum friction compensation value calculated from the circularity test may result in overcompensation on the relevant axis if it executes very short axial positioning movements (e.g. on infeeds in the µm range).

To improve accuracy in such cases too, it is possible to reduce the compensation amplitude for short traversing motions.

This weighting factor programmed in MD 32580: FRICT_COMP_INC_FACTOR automatically takes effect when friction compensation is activated (conventional QEC or QEC with neural networks) acting on all positioning movements that are made within an interpolation cycle of the control.

The input range is between 0 and 100% of the calculated compensation value.

**Control of learning process duration**

As described in previous sections, the duration of the learning process is dependent on several parameters. It is mainly dependent on the following values:

- Coarse quantization ($AA_QEC_COARSE_STEPS)
- Measuring time for determining the error criterion ($AA_QEC_MEAS_TIME_1 up to $AA_QEC_MEAS_TIME_3)
- Number of learning passes
- Detailed learning active [yes/no]?
- Fine quantization ($AA_QEC_FINE_STEPS) (only if “detailed learning = yes” is selected)
- Directional compensation active [yes/no]? ($AA_QEC_DIRECTIONAL)
- Duration of reversing movement

The setting “Detailed learning active = yes” causes a significant increase in the time required for learning. It must therefore only be used where precision requirements are high. It is necessary to check whether these requirements only apply to certain acceleration ranges. If so, detailed learning only needs to be performed section by section (see “Section-by-section learning y/n?”). The number of learning passes must be reduced in any case.

If the reference NC programs mentioned above are used with the recommended parameter values, the following times have been determined for the learning process time:

- Detailed learning not active: approx. 6.5 min
- Detailed learning active: approx. 13 min

## 2.6.6 Quick start-up

**Preparation for "Learning"**

1. Determining the optimum friction compensation time constant (MD 32540 FRICT_COMP_TIME) with conventional friction compensation.
2. **Enter the following machine data <u>without</u> power ON:**

| Machine data | Standard | Change to | Meaning |
|---|---|---|---|
| MD 19330<br>NC-CODE_CONF_NAME_TAB[8] | 0 | | Activate option "IPO_FUNKTION_MASK".<br>**Only with learn program "Polynomial"! Bit4 = 1** |
| MD 19300 COMP_MASK | 0 | | Set option |
| MD 32490 FRIC_COMP_MODE | 1 | 2 | "Type of friction compensation" neural QEC |
| MD 32500 FRIC_COMP_ENABLE | 0 | 0 | "Friction compensation active" for learning **"OFF"** |
| MD 32580 FRIC_COMP_INC_FACTOR | 0 | 0 | "Weighting factor of friction compensation value for short traversing motions" (mm increments) |
| MD 38010 MM_QEC_MAX_POINTS | 0 | 400 | "Selection of values for QEC" = $AA_QEC_FINE_STEPS * ($AA_QEC_COARSE_STEPS + 1) |
| MD 32620 FFW_MODE | 1 | 1 | Speed feedforward control |
| MD 32610 VELO_FFW_WEIGHT | 1 | 1 | Injection 100% |
| MD 32630 FFW_ACTIVATION_MODE | 1 | 0 | Feedforward control ON continuously |
| MD 32810 EQUIV_SPEEDCTRL_TIME | 0,004 | Initial value t_pos + n_setSm.* | Adjust equivalent time constant n control loop |

* t_position ... position control cycle (=basic system cycle * factor for position control cycle), n_setSm. ... speed setpoint smoothing (MD 1500 to 1521)

3. **Owing to re-allocation of memory (MD 38010), save the machine data with MMC100:**
   "Services" "Data Out" "Start-up data, NCK data" and if programmed, "LEC, measuring system error, sag and angularity error compensation tables" via PCIN. Execute a power ON Reset and then read in the saved data with PCIN and "Data IN". (= series start-up).
   **MMC102:**
   Save "SERIES STARTUP" and if programmed, "LEC, measuring system error, sag and angularity error compensation tables".
   Execute a power ON Reset and then read in the "Start-up" archive (saved data are loaded again).

4. **Copy the programs supplied on disk "MMC 100 TOOLBOX"into the NC (with archive!)**
   QECDAT.MPF
   QECSTART.MPF
   QECLRNP.SPF ("Polynomial" learning program)**or** QECLRNC.SPF ("Circle" learning program) is stored as QECLRN.SPF on the NC!
   With geometry axes, it is preferable to use the Circle learning program; for all other axes the Polynomial learning program only.
5. **Make the following program adjustments:**
   **–In part program QECDAT**
   Adjust friction compensation time constant if necessary (see para. 1)
   N1340 $AA_QEC_TIME_1[outNr,axNr] = 0.0xx

| | |
|---|---|
| N1040 def int numAxes = ..... | Enter the number of axes to be learned. |
| N1150 axisName[0] = ...... | Axis name Enter 1st axis. |
| N1160 axisName[1] = ...... | Axis name Enter 2nd axis. |
| N1170 axisName[2] = ...... | Axis name Enter 3rd axis. |
| N1180 axisName[3] = ...... | Axis name Enter 4th axis. |

   (For the "Circle" learning program, AX1 .. AX8 or the machine or channel axis name can be used as the axis name. However, only the channel axis name can be used for the "Polynomial" learning program).
   **–In part program QECSTART**

| | |
|---|---|
| N1080 def int numAxes = ..... | Enter the number of axes to be learned. |
| N1310 axisName[0] = ...... | Axis name Enter 1st axis. |
| N1320 axisName[1] = ...... | Axis name Enter 2nd axis. |
| N1330 axisName[2] = ...... | Axis name Enter 3rd axis. |
| N1340 axisName[3] = ...... | Axis name Enter 4th axis. |

   (For the "Circle" learning program, AX1 .. AX8 or the machine or channel axis name can be entered as the axis name. In contrast, only the channel axis name may be used for the "Polynomial" learning program).

### Executing "Learning" process

Start the following programs

- Select and start QECDAT.
  System variables are assigned.
- Select QECSTART, override 100% and start.
  The learning program runs for about 15 minutes involving approximately 30 cm traversing motions. If the message "REORG not possible" is displayed, it can be ignored. The message is displayed for about 10 seconds. It then disappears and the learning process continues with traversing motions.

### Activate QEC

| Machine data | Standard | Change to | Meaning |
|---|---|---|---|
| MD 32500<br>FRIC_COMP_ENABLE | 0 | 1 | Switch on "Friction compensation active" |

### "Circularity test"

**Use the "Circularity test"to check the result!**

**Save compensation data**

**Save the compensation data (QEC data are not saved with "SERIES START-UP"; can be selected in SW 4 and higher):**
**MMC100:**
Save with PCIN under SERVICES Data\Circle error compensation\All
**MMC102:**
Save the file under SERVICES NCK \ NC Active Data \ Quadrant Error Co \ Quadrant error comp-complete.ini. This file contains all compensation values.
Note: Change the "displayed name length" to "20" in SERVICES "System settings" "for display" to ensure that the whole name is visible.

## 2.7 Circularity test

**Function**

One of the purposes of the circularity test is to check the contour accuracy obtained by the friction compensation function (conventional or neural QEC). It works by measuring the actual positions during a circular movement and displaying the deviations from the programmed radius as a diagram (especially at the quadrant transitions).

**Procedure**

The circle contour for the axes involved is specified by an NC program. To simplify the circularity test as much as possible for the start-up engineer, an NC program is provided as a reference example for the circularity test motion (file QECTEST.MPF on the diskette with the basic PLC program). The start-up engineer must adapt this NC program to his application.

Several measurements must be made during the circularity test with different acceleration values to ascertain whether the learnt compensation characteristic (for neural QEC) or the defined compensation values (for conventional QEC) meet the requirements.

The circular movement can easily be made with different accelerations if you change the feedrate using the feedrate override switch without changing circular contour. The real feedrate must be taken into account in the measurement in the input field “feedrate”.

The circle radius chosen must be typical of machining operations on the machine (e.g. radius in the range 10 to 200 mm).

For the duration of the circular movement, the position actual values of the axes are recorded and stored in a “trace” in the passive file system. The circularity test is therefore purely a measuring function.

**Parameterization of circularity test**

You select in this menu the **names** or numbers of the axes with which the circle is traversed and whose actual position data must be recorded. No check is made to find out whether the selected axes match the axes programmed in the NC parts program.

The parameter settings in the input fields “Radius” and “Feed” must correspond to the values from the parts program that controls the circular motion of the axes, taking account of the feed override switch setting. No check is made to see whether the values in the parts program (including feedrate override) and the input values match.

The “Measuring time” display field shows the measuring time calculated from the “Radius” and “Feed” values for recording the position actual values during the circular movement.

If only parts of the circle can be represented (i.e. measuring time too short) the measuring time can be increased in the menu by reducing the feed value. This also applies if the circularity test is started from the stationary condition.

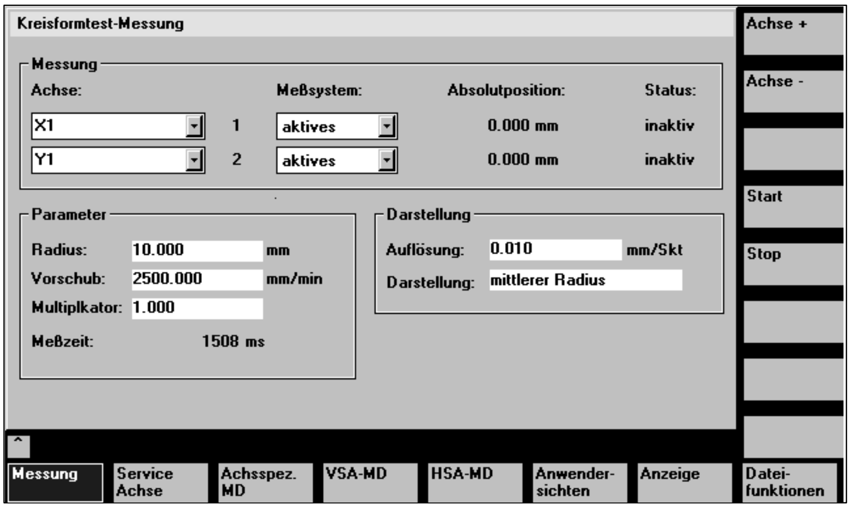



Fig. 2-32 Circularity test measurement menu

**Mode of representation**

The following parameter assignments for programming the mode of representation of measurement results can also be made:

- Display based on mean radius
- Display based on programmed radius
- Scaling of the diagram axes

If the measuring time calculated exceeds the time range that can be displayed from the trace buffers (maximum measuring time = position control cycle frequency * 2048), a coarser sampling rate is used for recording (n * position control cycle frequency), so that a complete circle can be displayed.

**Starting the measurement**

The operator must use an NC Start to start the parts program in which the circular motion for the selected axes is stored (AUTOMATIC or MDA operating mode).

The measuring function is started with the vertical soft key **Start**.

The sequence of operations (NC Start for part program and Start measurement) can be chosen by the user according to the application.

When the circularity test is active for the specified axes, the message “active” appears in the “Status” display field.

**Stopping the measurement**

The measurement can be stopped at any time by pressing the **Stop** key. Any incomplete measurement recordings are best displayed by selecting the **Display** soft key. There is no monitoring in this respect.

To allow direct access to the required controller parameters, the soft keys **Axis-specific MD, FDD-MD** and **MSD-MD** are displayed. The vertical soft keys **Axis+** and **Axis–** can be used to select the desired axis.

The “Service axis” display is displayed when you press the **Service Axis** soft key. The following service data are displayed here for commissioning of the friction torque compensation:

- QEC learning active yes/no?
- Current position and actual speed values

**Display**

When you press the **Display** soft key, the display switches to the graphical view of the recorded circle diagram.

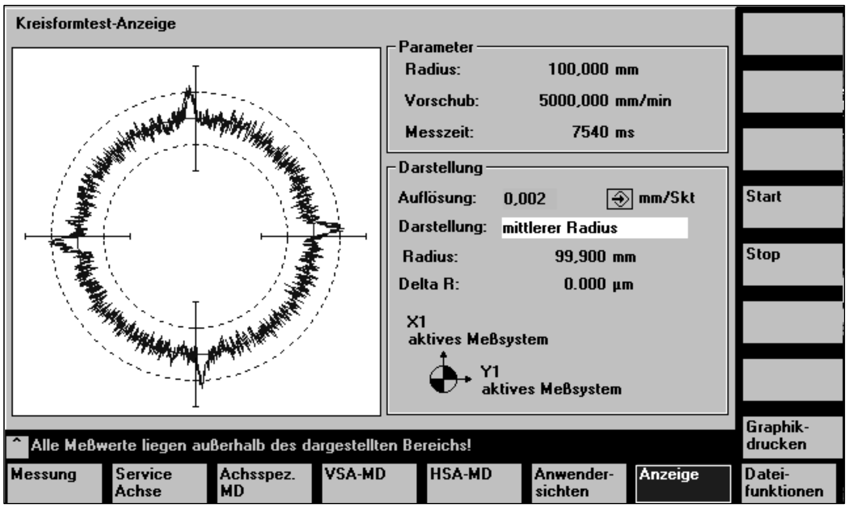



Fig. 2-33 Circularity test display menu

This screen displays the measurements of the two actual position values as a circle with the set resolution.

The programmed radius, the programmed feedrate and the measuring time derived from these values are also displayed for documentation purposes (for subsequent storage of the measured circle characteristics in file format).

The operator can enter a finer scale for the diagram axes in the **Resolution** input field, e.g. in order to emphasize the transitions at the quadrants. The circle diagram is refreshed with the new resolution when you press the **Display** softkey.

**File functions**

The displayed measurement results and the parameter settings can be stored as a file on the MMC by selection of softkey **File functions**.

**Printer settings**

The basic display for selecting a printer (Fig. 10-15) can be called by means of soft keys **MMC Printer selection**.

The toggle key is used to define whether the displayed graphic is to be output directly on the printer or transferred to a bit map file after softkey **Print graph** is selected.

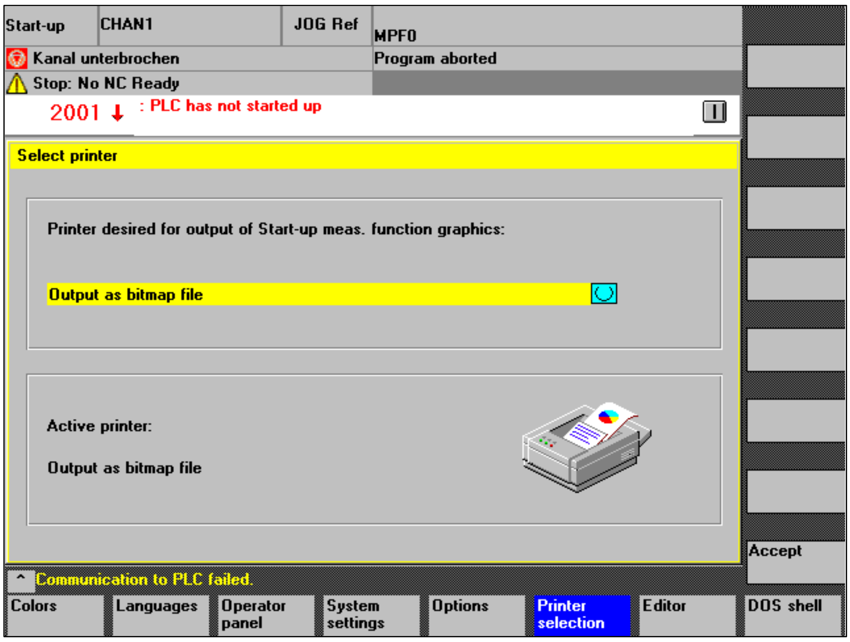



Fig. 2-34 Basic screen for printer selection

**Direct output on printer**

The printer must be set up under MS-Windows.

"Output on printer" is set in the selection field.
After selection of the softkey labeled **Print graph**, the displayed graph is output on the connected printer.

**Output as bitmap file**

The graphic is stored in a bitmap file (*.bmp).

"Output as bitmap file" is chosen in the printer settings selection field.
When the softkey labeled **Print graph** is selected, the screen form for assigning a file name appears in the "Circularity test display". A new file name can be entered or an existing file name selected for overwriting in the drop-down list.

Softkey **Ok** is selected to store the file.
Softkey **Abort** is selected to return to the current graphic display.

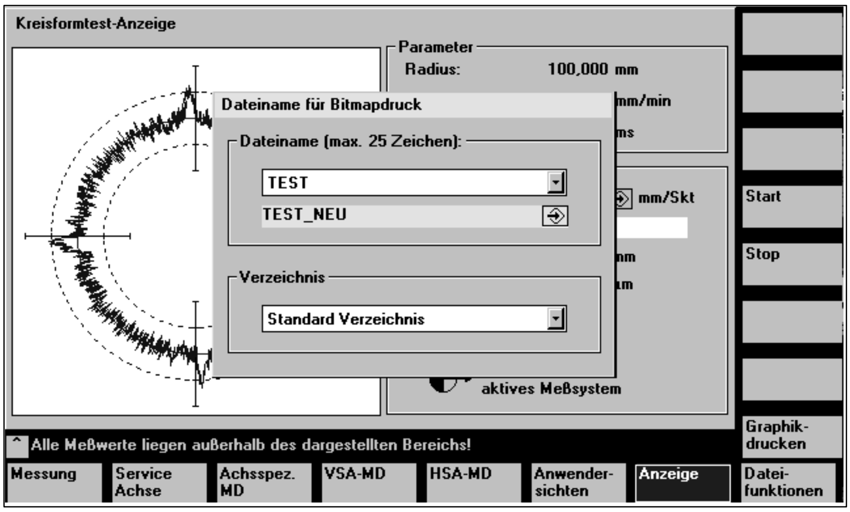



Fig. 2-35 Assignment of file name for output in a bitmap file.

## 2.7.1 Neural quadrant error compensation, quick start-up

**Quick start-up "Neural Quadrant Error Compensation"with parabolic/circular movements on MMC102/103/MMC100**

**Preparation for "Learning"**

The friction compensation time constant (MD 32540 FRICT_COMP_TIME) is calculated first by means of conventional friction compensation.

Table 2-2 Enter the following machine data **without** power ON

| Machine data | Standard | Change to | Meaning |
|---|---|---|---|
| MD 19330 IPO_FUNCTION_MASK | 0 | 8 | Activate "Polynomial interpolation" option.<br>**For polynomial only! Bit 4=1** |
| MD 19300 COMP_MASK | 0 | 8 | Option "Neural QEC", bit 4 = 1 |
| MD 32490 FRIC_COMP_MODE | 1 | 2 | "Type of friction compensation" neural QEC |
| MD 32500 FRIC_COMP_ENABLE | 0 | 0 | "Friction compensation active" for learning "OFF" |
| MD 32580 FRIC_COMP_INC_FACTOR | 0 | 0 | "Weighting factor of friction compensation value for short traverse motions" (μm increments) |
| MD 38010 MM_QEC_MAX_POINTS | 0 | 400 | "Selection of values for QEC" = $AA_QERC_FINE_STEPSA * ($AA_QEC_COARSE_STEPS + 1) |
| MD 32620 FFW_MODE | 1 | 1 | Speed feedforward control |
| MD 32610 VELO_FFW_WEIGHT | 1 | 1 | Injection 100% |
| MD 32630 FFW_ACTIVATION_MODE | 1 | 0 | Feedforward control ON continuously |
| MD 32810 EQUIV_SPEEDCTRL_TIME | 0,004 | t_pos.+ n_setSm. *) | Adjust equivalent time constant n control loop |

*) t_position ... position control cycle (=basic system cycle * factor for position control cycle), n-setSm. ... speed setpoint filter (MD 1500 ... 1521)

Owing to re-allocation of memory (MD 38010), save the machine data with:

**MMC100:** Save “Services”, “Data OUT”, “Start-up data, NCK data” and, if programmed, measuring system error and sag/angularity compensation tables via PCIN, execute a power ON-Reset and then read in the saved data with PCIN and “Data IN” (=series machine start-up).

**MMC102/103:** Save “SERIES START-UP” and, if programmed, measuring system error and sag/angularity compensation tables, execute power ON-Reset and read in “Start-up” archive (saved data are reloaded).

**Copy the programs supplied on disk “MMC 100 TOOLBOX”into the NC (with archive!)**

QECDAT.MPF
QECSTART.MPF
QECLERNP.SPF (“Polynomial” learning program) or QECLRNC.SPF (“Circle” learning program) is stored as QECLRN.SPF on the NC!

**Adapt the following programs:**

- **In parts program QECDAT**

  N1040 def int numAxes=... Enter the number of axes to be learned

  N1150 axisName[0] Enter axis name of 1st axis.
  N1160 axisName[1] Enter axis name of 2nd axis.
  N1170 axisName[2] Enter axis name of 3rd axis.
  N1180 axisName[3] Enter axis name of 4th axis.
  (AX1 .. AX8 or the machine or channel axis name can be used as an axis name for the “Circle” learning program. In contrast, only the channel axis name may be used for the “Polynomial” learning program).

- **In part program QECSTART**

  N1080 def int numAxes=... Enter the number of axes to be learned

  N1310 axisName[0] Enter axis name of 1st axis.
  N1320 axisName[1] Enter axis name of 2nd axis.
  N1330 axisName[2] Enter axis name of 3rd axis.
  N1340 axisName[3] Enter axis name of 4th axis.
  (AX1 .. AX8 or the machine or channel axis name can be used as an axis name for the “Circle” learning program. In contrast, only the channel axis name may be used for the “Polynomial” learning program).

**Execute LEARN process**

Select and start QECDAT

System variables are assigned.

Select QECSTART 100% override and start

The learning program runs for about 15 minutes involving approximately 30 cm traversing motions. The message “REORG not possible” can be ignored if it occurs. The message is displayed for about 10 seconds. It then disappears and the learning process continues with traversing motions.

### Activate QEC

| Machine data | Standard | Change to | Meaning |
|---|---|---|---|
| MD 32500 FRIC_COMP_ENABLE | 0 | 1 | Switch on “Friction compensation active” |

**Use the “Circularity test” to check the result!**

Save compensation data (QEC data are not included in back-up with “SERIES START-UP):

**MMC100:** Save with PCIN under SERVICES\Data\Circle error compensation\All.

**MMC102/103:** Save the file under SERVICES\NCK\NC Active Data\Quadrant Error Co\Quadrant error comp-complete.ini. This file contains all compensation values.

---

**Note**

Change the “displayed name length” to “20” in SERVICES “System settings” “for display” to ensure that the whole name is visible.

---

## 2.8 Electronic weight compensation (vertical axis)

**Prerequisite**

This function is available only for use in conjunction with SIMODRIVE 611 digital drives.

---

**Note**

The “Electronic weight compensation” functionality is not currently available for the combination SINUMERIK 840Di and SIMODRIVE 611 universal drive.
The parameters required for the function cannot be transferred to the drive via the PROFIBUS-DP.

---

## 2.8.1 Electronic counterweight function

**Axis without electronic weight compensation**

In the case of weight-bearing axes without weight compensation, the vertical axes drop when the brake is released and the system responds as follows:

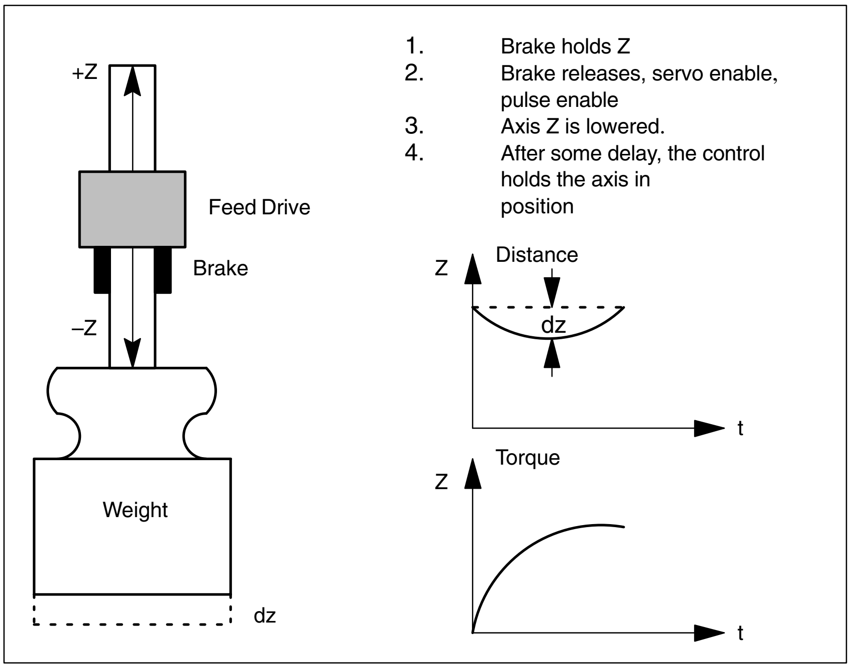



Fig. 2-36 Lowering of a vertical axis without electronic weight compensation

The amount the axis (Z) is lowered increases in proportion to the speed controller reset time in MD 1409: SPEEDCTRL_INTEGRATOR_TIME_1 set with the SIMODRIVE 611 digital. Through activation of the electronic weight compensation function, it is possible to minimize the amount by which the axis is lowered.

**Activation**

The function is activated by setting axis-specific MD 32460:TORQUE_OFFSET to a value **other than zero** and made operative on the next RESET or POWER ON or via softkey key “Activate MD”.

**Axis with electronic weight compensation**

The electronic weight compensation function prevents weight-bearing axes from sagging when the control is switched on. After releasing the brake, the constant weight compensation torque maintains the position of the vertical axis. This process is illustrated in the figure below.

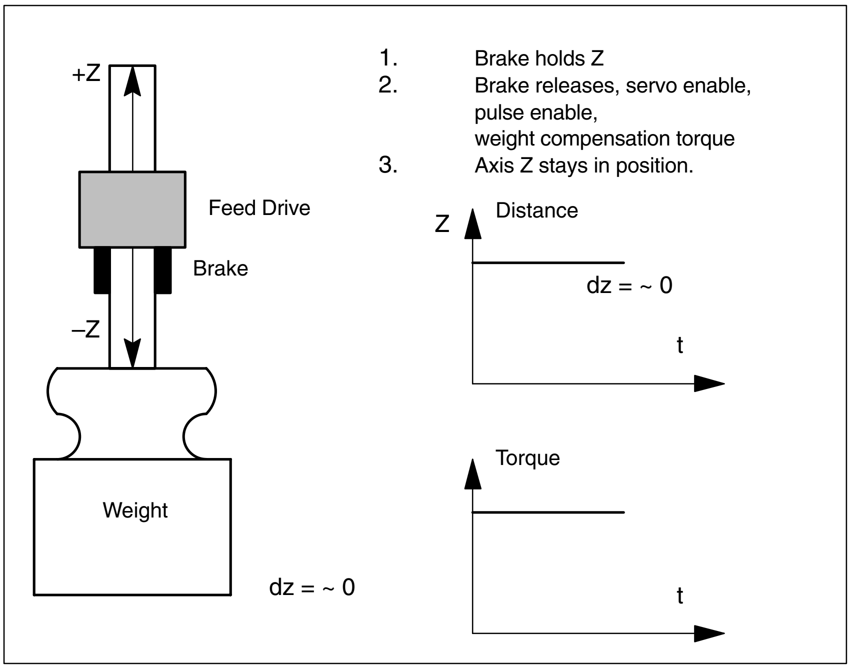



Fig. 2-37 Lowering of a vertical axis with electronic weight compensation

---

**Note**

This function is available only in conjunction with SIMODRIVE 611 digital.

---

**Switching off**

The electronic weight compensation function is deactivated with setting

MD 32460:TORQUE_OFFSET = 0.

The deselection takes effect after the next RESET or power ON or on selection of softkey “Activate MD”.

### 2.8.2 Effect on electronic counterweight function of rebooting from HMI

**Introduction**

HMI is capable of booting the NCK for the purpose, for example, of activating machine data. The result of this reboot would be that vertical axes would drop a short distance. Use can be made of the "**Reboot Management**" function to avoid this problem.

When the management function is appropriately set, the NCK permits the NCK and PLC to be powered down with a delay and also signals that shutdown is imminent. This allows enough time for brake activation to prevent vertical axes from dropping.

The Reboot Management function works only in conjunction with controlled Power On via the HMI. A power failure or hardware reboot does not activate the Reboot Management function.

**REBOOT sequence**

The HMI initiates an NCK and PLC reboot using PI service "_N_IBN_SS".

NCK immediately activates alarm 2900 in response to the PI service. Machine data MD 10088: REBOOT_DELAY_TIME then specifies the time period allowed by the NCK between the PI service and initiation of reboot. This time delay can be used to activate **mechanical axis brakes**.

**Reactions to alarm 2900**

1. The following VDI signals are **canceled**, i.e. set to zero.
   Mode Group ready
   DB 11 DBB 6 Bit 3 (all mode groups)
   Channel ready
   DB 21 DBB 36 Bit 5 (all channels)
   Axis ready
   DB 31 DBB 61 Bit 2 (all axes)

2. The "Ready" message at relay contacts 72 73.1 73.2 74 is **not** reset.
3. The NCK brakes along the current limit.
   See also
   MD 36610: AX_EMERGENCY_STOP_TIME and
   MD 36620: SERVO_DISABLE_DELAY_TIME.
   **Note:** The NCK deactivates the position control after SERVO_DISABLE_DELAY_TIME.
4. The following VDI signals remain at 1.
   NC ready
   DB 10 DBB 108 Bit 7

Machine data

MD 11410: SUPPRESS_ALARM_MASK (BIT20) can be used to suppress alarm 2900, but the NCK still initiates the same reactions.

As alarm 2900 deactivates the axis position control, this alarm must be configured to effect **application of the mechanical brakes 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 management function is activated when MD 10088: REBOOT_DELAY_TIME is set to a value other than zero.

**Evaluation with a system variable**

System variable $AN_REBOOT_DELAY_TIME can be read in a synchronous action. A value higher than zero indicates that the reboot request initiated by the HMI has been issued and how much time (in seconds) the NCK will allow until reboot (Power Off followed by Power On). The user can detect an imminent reboot by reading a synchronous action and react accordingly (e.g. with “Safe Standstill” in a Safety Integrated application). $AN_REBOOT_DELAY_TIME is 0.0 as long as the HMI has not initiated a reboot request.
See also /PGA1/, List of System Variables.

### 2.8.3 Electronic weight compensation with travel to fixed stop

**SIMODRIVE 611 digital up to SW 5.1**

**With NC SW 6** and SW 5.1 SIMODRIVE 611 digital and earlier, both functions "electronic weight compensation" and "travel to fixed stop" can be used simultaneously, but the following special points should be noted in this respect:

**Interaction with traverse against fixed stop**

The electronic weight compensation may **not** be used to offset the zero point for the fixed stop torque or fixed stop force as it is unsuitable for this purpose.

- If, for example, the axis requires 30% weight compensation in a case where 40% fixed stop torque is programmed in the same direction, then the actual torque with which the axis presses against the fixed stop only corresponds to 10% of rated torque.
- If 40% fixed stop torque is programmed in the other direction (in the opposite direction to weight compensation, i.e. in direction in which axis would drop) in the same situation described above, then the actual torque with which the axis presses against the fixed stop corresponds to 70% of rated torque.
- If the axis needs, for example, a weight counterbalance of 30%, then it is **not** possible for the axis to approach a fixed stop if less than 30% fixed stop torque is programmed. The drive torque would be limited so severely that the axis could no longer be kept under control and would drop!

These characteristics of the traverse against fixed stop function with vertical axes are determined by the available options for torque limitation in the drive. They are neither improved nor impaired by the weight compensation function.

**SIMODRIVE 611 digital SW 5.1 and higher**

For travel to fixed stop **with NC-SW 6** or SIMODRIVE 611 digital SW 5.1 and higher, it is also possible to set a torque limit in the NC that is smaller than the weight of the drive. In doing this, a torque/force limit is evaluated by the NC. This torque/force limit is operative in addition to the limits set in the drive

- Current,
- Force/torque,
- Power, pullout power
- Setup mode

**Required adjustments**

The torque/force limit is entered for the different drive types in the drive machine data provided for this purpose.

| Drive machine data | Drive type | Meaning |
|---|---|---|
| MD 1192: TORQUE_LIMIT_WEIGHT | FDD/MSD | The torque corresponding to the force due to weight |
| MD 1192: FORCE_LIMIT_WEIGHT | 1FN1 and 1FN3 | The force due to weight with linear motors |
| MD 5231: FORCE_LIMIT_WEIGHT | HLD Module | The force due to weight with hydraulic drives |

## Relationships for adjustment

**Limit values of NC symmetrical to SIMODRIVE**

Machine data MD 1192 uses the same unit in percent (%) as the NC machine data MD 32460: TORQUE_OFFSET[n] additional torque for electronic counterweight. The torque/force limit of the NC therefore acts symmetrically above and below this weight torque/force. The resulting static torque/static force relationship is as follows:

**Formula**

$M_0 = k_T * I_0$ for synchronous motors (13)

or rated torque:

$M_{rated}$ for induction motors

$c_T$ = Torque constant [Nm/A]
$I_0$ = Current at zero speed [A]

**Manual NC format adjustment with SIMODRIVE**

In order to facilitate the setting procedure MD 1728: DESIRED_TORQUE displays the current torque/force setpoint in the same format as in MD 1192 and MD 32460: TORQUE_OFFSET[n].

If only the force due to weight is active, the value can be read and MD 1192 and MD 32460: TORQUE_OFFSET[n] transferred. If the value of the force due to weight is greater than the torque/force limit of the NC, then the upper and lower torque/force limit has the same leading sign. If the force due to weight is entered incorrectly, it can result in constant acceleration once the NC force limit is reached!

With SW 5.1 and higher, the torque limit for setup mode MD 1239: TORQUE_LIMIT_FOR_SETUP and the force limit MD 1239: FORCE_LIMIT_FOR_SETUP also act symmetrically at the force due to weight. The minimum is selected from the limit of NC and setup mode if setup mode is active.

**With SW 5.1 and higher** for SIMODRIVE 611 digital, the limits of the NC and setup mode are no longer included in

- the reference values for ramp-function generator follow-up
- the capacity utilization and
- the M<Mx signaling function.

■

# Notes

# Supplementary Conditions 3

## 3.1 Availability

The individual compensation types are:

- Backlash compensation
- Leadscrew error and measuring system compensation
- Multi-dimensional beam sag compensation
- Manual quadrant error compensation
- Automatic quadrant error compensation (neural network)
- Temperature compensation
- Automatic drift compensation for analog speed setpoints
- Electronic weight compensation

**“Backlash compensation” function**

This function is available for:

- SINUMERIK 840D with NCU 571/572/573 with SW 1 and higher

**“Leadscrew error and measuring system error compensation” function**

This function is available for:

- SINUMERIK 840D with NCU 571/572/573 with SW 1 and higher

**“Multi-dimensional beam sag compensation” function**

The function is an option and is available for

- SINUMERIK 840D with NCU 571/572/573 with SW 2 and higher

The function is contained in the export version 840DE with restricted functionality; it is not contained in the FM-NC, 810DE (SW 3.1 and lower).

- The function is available for the SINUMERIK 810DE in SW 3.2 and higher.

**"Quadrant error compensation by operator input" function**

This function is available for:

- SINUMERIK 840D with NCU 571/572/573 with SW 1 and higher

**"Automatic quadrant error compensation" function**

The function is an option and is available for:

- SINUMERIK 840D with NCU 571/572/573 with SW 2 and higher

**"Temperature compensation" function**

The function is an option and is available for:

- SINUMERIK 840D with NCU 571/572/573 with SW 1 and higher
- SINUMERIK FMNC with NCU 570 with SW 1 and higher

**"Electronic weight compensation" function**

This function is available for:

- SINUMERIK with NCU 571/572/573, SW 3 and higher, in conjunction with SIMODRIVE 611D.

■

# Data Descriptions (MD, SD) 4

## 4.1 Description of machine data

### 4.1.1 General machine data

| | | | | | |
|---|---|---|---|---|---|
| **10082**<br>MD number | **CTRLOUT_LEAD_TIME**<br>Shift of setpoint transfer time | | | | |
| Default setting: 0.0 | | Minimum input limit: 0.0 | | Maximum input limit: 100.0 | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: % |
| Data type: DOUBLE | | | | Applies from SW: 2 | |
| Meaning: | Lead time for output of the speed setpoints.<br>The larger the value entered, the sooner the drive accepts the speed setpoints.<br>• 0 % setpoints are transferred at the beginning of the next position control cycle.<br>• 50 % setpoints are already transferred after execution of half of the position control cycle.<br>A reasonable lead time can only be determined by measuring the maximum position control calculating time.<br>In the machine data 10083: CTRLOUT_LEAD_TIME_MAX suggests a value measured by the control. As this is a net value, it is advisable for the user to provide for a safety allowance of, for example, 5%.<br>If lead times that are too high are input, this can cause output of drive alarm 300506.<br>The input value is rounded to the next speed controller pulse rate in the drive.<br>If the speed controller pulse rates of the drives are different, changing the value will not necessarily lead to the same degree of controller improvement for all configured drives.<br>**Note:** This MD is only important for axes with digital drives. | | | | | |
| Related to .... | MD 10083: CTRLOUT_LEAD_TIME_MAX | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **10083**<br>MD number | **CTRLOUT_LEAD_TIME_MAX**<br>Maximum permissible setting for shift of setpoint transfer time | | | | |
| Default setting: 100.0 | | Minimum input limit: 0.0 | | Maximum input limit: 100.0 | |
| Changes effective after NEW_CONF | | | Protection level: 2 / 7 | | Unit: % |
| Data type: DOUBLE | | | | Applies from SW: 4 | |
| Meaning: | Maximum permissible lead time for output of the speed setpoints.<br>MD 10083 represents a setting aid for MD 10082.<br>The displayed value can be directly transferred to MD 10082, taking the safety allowance into account.<br>The permissible lead time is determined from the maximum measured computing time required by the position controller. It decreases as the position controller's computing time requirements increase.<br>By reducing the position control cycle via MD 10060 or 10050, you can reduce the permissible lead time.<br>The lead time is measured during the entire operating life. The displayed value can only be increased by manual input.<br>If the specified lead time is greater than the permissible one (e.g. 100%), then it is automatically determined again.<br>**Note:** This MD is only important for axes with digital drives. | | | | | |
| Related to .... | MD 10050: SYSCLOCK_CYCLE_TIME (system clock time)<br>MD 10060: POSCTRL_SYSC LOCK_TIME_RATIO (factor for position control cycle)<br>MD 10082: CTRLOUT_LEAD_TIME | | | | |

<table>
<tr><td>10088<br>MD number</td><td colspan="4">REBOOT_DELAY_TIME<br>Reboot delay</td></tr>
<tr><td>Default setting: 0.2</td><td colspan="2">Minimum input limit: 0.0</td><td colspan="2">Maximum input limit: 1.0</td></tr>
<tr><td colspan="2">Changes effective immediately</td><td colspan="2">Protection level: 3 / 3</td><td>Unit: s</td></tr>
<tr><td colspan="3">Data type: DOUBLE</td><td colspan="2">Applies from SW: 7.2</td></tr>
<tr><td>Meaning:</td><td colspan="4">The reboot operation which follows PI “N_IBN_SS” is delayed by the time REBOOT_DELAY_TIME.<br>The suppressible NOREADY alarm 2900 is activated immediately by PI “_N_IBN_SS”.<br><br>If REBOOT_DELAY_TIME is shorter than the time set in MD 36620: SERVO_DISABLE_DELAY_TIME for a particular axis,<br>then the axis will be braked while the REBOOT_DELAY_TIME is active and the servo enable signal will then be canceled, in other words, the time SERVO_DISABLE_DELAY_TIME is n_o_t fully applied.<br><br>When REBOOT_DELAY_TIME = 0.0, alarm 2900 is not activated and no reboot delay is applied.</td></tr>
</table>

<table>
<tr><td>18342<br>MD number</td><td colspan="4">MM_CEC_MAX_ POINTS[t]<br>Maximum number of interpolation points for beam sag compensation [table t]</td></tr>
<tr><td>Default setting: 0</td><td colspan="2">Minimum input limit: 0</td><td colspan="2">Maximum input limit: 2000</td></tr>
<tr><td colspan="2">Changes effective after POWER ON</td><td colspan="2">Protection level: 2 / 4</td><td>Unit: –</td></tr>
<tr><td colspan="3">Data type: DWORD</td><td colspan="2">Applies from SW: 2.1</td></tr>
<tr><td>Meaning:</td><td colspan="4">For beam sag compensation, the number of required interpolation points must be defined for every compensation table [t].<br><br>where: [t] = Index of compensation table<br><br>with ($0 \leq t \leq 2$ * maximum number of axes)<br>i.e. t = 0: 1. compensation table<br>t = 1: 2. compensation table etc.<br><br>The necessary number can be calculated from the defined parameters as follows (see Subsection 2.3.3):<br>$$\text{MM\_CEC\_MAX\_POINTS[t]} = \frac{\text{\$AN\_CEC\_MAX[t]}-\text{\$AN\_CEC\_MIN[t]}}{\text{\$AN\_CEC\_STEP[t]}} + 1$$<br>$AN_CEC_MIN [t] Initial position (system variable)<br>$AN_CEC_MAX [t] End position (system variable)<br>$AN_CEC_STEP [t] Distance between interpolation points (system variable)<br><br>When selecting the number of interpolation points and the distance between them the resulting size of the compensation table and the resulting required memory capacity in the non-volatile user memory must be noted. 8 bytes are required for every compensation value (interpolation point).<br><br>If the value 0 is entered, no memory is reserved for the table; i.e. the table does not exist and the function cannot therefore be activated.</td></tr>
<tr><td>Special cases, errors, ......</td><td colspan="4">Caution!<br><br>When MD 18342: MM_CEC_MAX_POINTS[t] is changed the non-volatile NC user memory is automatically reallocated on power ON. This deletes all the user data in the non-volatile user memory (e.g. drive and MMC machine data, tool offsets, parts programs etc.).</td></tr>
<tr><td>Related to ....</td><td colspan="2">SD 41300: CEC_TABLE_ENABLE[t]</td><td colspan="2">Enable evaluation of beam sag compensation table [t]</td></tr>
<tr><td>References</td><td colspan="4">/FB/, S7, “Memory Configuration”</td></tr>
</table>

## 4.1.2 Axis-specific machine data

| | | | | |
|---|---|---|---|---|
| **32450** MD number | **BACKLASH[n]** Backlash | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective after NEW_CONF | | | Protection level: 2 | Unit: mm or degrees |
| Data type: DOUBLE | | | Applies from SW: 1.1 | |
| Meaning: | Backlash between the positive and the negative direction of travel.<br>The compensation value input is<br>• positive, if the encoder leads the machine part (normal case)<br>• negative, if the encoder lags behind the machine part.<br>If zero is entered backlash compensation is deactivated.<br>Backlash compensation is always activated after reference point approach in all modes.<br>The index [n] has the following coding:<br>[encoder no.]: 0 or 1 | | | |
| Special cases, errors, ...... | If there is a second measuring system, a separate backlash value must be entered for this measuring system. | | | |
| Related to .... | MD: NUM_ENC (number of measuring systems)<br>MD: ENC_CHANGE_TOL (maximum tolerance for position actual-value switchover) | | | |

| | | | | |
|---|---|---|---|---|
| **32452** MD number | **BACKLASH_FACTOR[n]** Weighting factor for backlash | | | |
| Default setting: 1.0 | | Minimum input limit: 0.01 | | Maximum input limit: 100.0 |
| Changes effective after NEW_CONF | | | Protection level: 2 / 7 | Unit: – |
| Data type: DOUBLE | | | Applies from SW: 5.1 | |
| Meaning: | Weighting factor for backlash<br>This machine data enables the backlash entered in MD 32450: BACKLASH to be changed as a function of a parameter set, e.g. in order to take account of gear-stage-specific backlash. | | | |
| Related to .... | MD 32450: BACKLASH[n] | | | |

| | | | | |
|---|---|---|---|---|
| **32460** MD number | **TORQUE_OFFSET** Additional torque for electronic weight compensation | | | |
| Default setting: 0 | | Minimum input limit: –100 | | Maximum input limit: 100 |
| Changes effective after NEW_CONF | | | Protection level: 2 / 7 | Unit: % |
| Data type: DOUBLE | | | Applies from SW: 3.1 | |
| Meaning: | The additional torque for the electronic weight compensation is entered in the % block of the static torque (calculated from MD1113 x MD1118). It is immediately effective when the current controller is activated. Vertical axes are thus prevented from sagging when the controller enabling signal is set, particularly when the speed controller reset time setting is high.<br><br>100% corresponds to the static torque of the axis drive.<br><br>With the speed controller deactivated, a positive value would move the drive in a positive traversing direction (see also MD 32100: AX_MOTION_DIR for further details).<br><br>If, therefore, the positive traversing direction is upwards (axis is raised), then a positive value must be entered for the weight compensation.<br>Conversely, a positive traversing direction downwards would call for a negative value.<br><br>MD is effective only for SIMODRIVE 611D drive systems. | | | |
| Special cases, errors, ...... | See Interaction with “Traverse against fixed stop” function | | | |
| Related to .... | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **32490**<br>MD number | **FRICT_COMP_MODE**<br>Friction compensation mode | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 2 | |
| Changes effective after NEW_CONF | | | Protection level: 2/4 | | Unit: – |
| Data type: BOOLEAN | | | | Applies from SW: 2.1 | |
| Meaning: | 0: No friction compensation<br>1: Friction compensation with const. feedforward value or adaptive characteristic<br>2: Friction compensation with learnt characteristic via neural network | | | | |
| Related to .... | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **32500**<br>MD number | **FRICT_COMP_ENABLE**<br>Friction compensation active | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after NEW_CONF | | | Protection level: 2 / 4 | | Unit: – |
| Data type: BOOLEAN | | | | Applies from SW: 1.1 | |
| Meaning: | 1: The axis is enabled for 'friction compensation' and therefore injection of the friction compensation values.<br>Quadrant errors on circular contours can be compensated with 'friction compensation'.<br>Axial MD 32490: FRICT_COMP_MODE "friction compensation type" defines whether "friction compensation with constant injected value" or "quadrant error compensation with neural networks" is selected.<br>In the case of neural networks, the machine data should first be set to "1" when a valid characteristic has been "learnt". During the learning phase, the compensation values are injected independently of the contents of this machine data.<br>0: Friction compensation is not enabled for this axis. No friction compensation values are injected. | | | | |
| Related to .... | MD 32490: FRICT_COMP_MODE<br>MD 32510: FRICT_COMP_ADAPT_ENABLE<br>MD 32520: FRICT_COMP_CONST_MAX<br>MD 32540: FRICT_COMP_TIME<br>MD 38010: MM_QEC_MAX_POINTS | | | Friction compensation type<br>Friction compensation adaptation active<br>Maximum friction compensation value<br>Friction compensation time constant<br>Number of interpolation points for quadrant error compensation with neural networks | |

| 32510<br>MD number | FRICT_COMP_ADAPT_ENABLE [n]<br>Friction compensation adaptation active [setpoint branch]: 0 |
|---|---|
| Default setting: 0 \| Minimum input limit: 0 \| Maximum input limit: 1 | |
| Changes effective after NEW_CONF \| Protection level: 2 \| Unit: – | |
| Data type: BOOLEAN \| Applies from SW: | |
| Meaning: | 1: Friction compensation with amplitude adaptation is enabled for the axis.<br>With 'friction compensation' quadrant errors on circular contours can be compensated. Often, the injection amplitude of the friction compensation value is not constant over the entire acceleration range. In this case, for high accelerations a smaller compensation value must be injected than for small accelerations to achieve optimum friction compensation.<br>The parameters of the adaptation curve (see Fig. 2-14) must be determined and entered in the machine data.<br>0: Friction compensation with amplitude adaptation must be enabled for the axis. |
| MD irrelevant for ...... | MD 32500: FRICT_COMP_ENABLE = 0<br>MD 32490: FRICT_COMP_MODE = 2 (neural QEC) |
| Related to .... | MD 32500: FRICT_COMP_ENABLE — Friction compensation active<br>MD 32520: FRICT_COMP_CONST_MAX — Maximum friction compensation value<br>MD 32530: FRICT_COMP_CONST_MIN — Minimum friction compensation value<br>MD 32550: FRICT_COMP_ACCEL1 — Adaptation acceleration value 1<br>MD 32560: FRICT_COMP_ACCEL2 — Adaptation acceleration value 2<br>MD 32570: FRICT_COMP_ACCEL3 — Adaptation acceleration value 3<br>MD 32540: FRICT_COMP_TIME — Friction compensation time constant |

| 32520<br>MD number | FRICT_COMP_CONST_MAX [n]<br>Maximum friction compensation value [setpoint branch]: 0 |
|---|---|
| Default setting: 0 \| Minimum input limit: 0 \| Maximum input limit: plus | |
| Changes effective after NEW_CONF \| Protection level: 2 \| Unit: mm/min | |
| Data type: DOUBLE \| Applies from SW: | |
| Meaning: | MD 32520: FRICT_COMP_CONST_MAX the magnitude of the (maximum) injection amplitude of the friction compensation value is defined.<br>This value is injected over the entire acceleration range for friction compensation without adaptation.<br>In the case of friction compensation with adaption, this value is merely applied in the acceleration range B2 of the adaptation characteristic (see Chapter 2, Subsection "Conv. friction friction compensation"). |
| MD irrelevant for ... ... | MD 32500: FRICT_COMP_ENABLE = 0<br>MD 32490: FRICT_COMP_MODE = 2 (neural QEC) |
| Related to .... | MD 32500: FRICT_COMP_ENABLE — Friction compensation active<br>MD 32510: FRICT_COMP_ADAPT_ENABLE — Friction compensation adaptation active<br>MD 32530: FRICT_COMP_CONST_MIN — Minimum friction compensation value<br>MD 32550: FRICT_COMP_ACCEL1 — Adaptation acceleration value 1<br>MD 32560: FRICT_COMP_ACCEL2 — Adaptation acceleration value 2<br>MD 32570: FRICT_COMP_ACCEL3 — Adaptation acceleration value 3<br>MD 32540: FRICT_COMP_TIME — Friction compensation time constant |

| | | | | | |
|---|---|---|---|---|---|
| **32530**<br>MD number | **FRICT_COMP_CONST_MIN [n]**<br>Minimum friction compensation value [setpoint branch]: 0 | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after NEW_CONF | | | Protection level: 2 | | Unit: mm/min |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | |
| Meaning: | The minimum friction compensation value is needed only if “Friction compensation with adaptation” is active (MD 32510: FRICT_COMP_ADAPT_ENABLE = 1).<br>The friction compensation amplitude entered in FRICT_COMP_CONST_MIN is applied in the acceleration range B4 ($a \geq a_3$) of the adaptation characteristic (see Chapter 2, Subsection “Conv. friction compensation”). | | | | |
| MD irrelevant for ... ... | MD 32510: FRICT_COMP_ADAPT_ENABLE = 0<br>MD 32490: FRICT_COMP_MODE = 2 (neural QEC) | | | | |
| Special cases, ...... | In exceptional cases, the value programmed for FRICT_COMP_CONST_MIN may even be higher than the setting for MD 32520: FRICT_COMP_CONST_MAX. | | | | |
| Related to .... | MD 32500: FRICT_COMP_ENABLE Friction compensation active<br>MD 32510: FRICT_COMP_ADAPT_ENABLE Friction compensation adaptation active<br>MD 32520: FRICT_COMP_CONST_MAX Maximum friction compensation value<br>MD 32550: FRICT_COMP_ACCEL1 Adaptation acceleration value 1<br>MD 32560: FRICT_COMP_ACCEL2 Adaptation acceleration value 2<br>MD 32570: FRICT_COMP_ACCEL3 Adaptation acceleration value 3<br>MD 32540: FRICT_COMP_TIME Friction compensation time constant | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **32540**<br>MD number | **FRICT_COMP_TIME[n]**<br>Friction compensation time constant [setpoint branch]: 0 | | | | |
| Default setting: 0.015 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after NEW_CONF | | | Protection level: 2 | | Unit: s |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | |
| Meaning: | Time constant over which the friction compensation value is injected (decay time of the compensation setpoint pulse).<br>Deviations at the quadrant transitions are not only influenced by the injection amplitude but also by a change in the friction compensation time constant (see Subsection 2.5.3). | | | | |
| MD irrelevant for ...... | MD 32500: FRICT_COMP_ENABLE = 0 | | | | |
| Related to .... | MD 32500: FRICT_COMP_ENABLE Friction compensation active<br>MD 32520: FRICT_COMP_CONST_MAX Maximum friction compensation value | | | | |

| | | | | |
|---|---|---|---|---|
| **32550**<br>MD number | **FRICT_COMP_ACCEL1 [n]**<br>Adaptation acceleration value 1 [setpoint branch]: 0 | | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after NEW_CONF | | Protection level: 2 | | Unit: $m/s^2$ |
| Data type: DOUBLE | | | Applies from SW: 1.1 | |
| Meaning: | The adaptation acceleration value is only required if “Friction compensation with adaptation” is active.<br>The adaptation acceleration values 1 to 3 are interpolation points for defining the adaptation curve. The adaptation curve is subdivided into four ranges in which different friction compensation values apply.<br>Range B1 is defined by FRICT_COMP_ACCEL1 ($a_1$) (see Chapter 2, Subsection “Conv. friction compensation”).<br>For the injection amplitude within range B1 the following applies:<br>$\Delta n = \Delta n_{max} * a / a_1$ for $a < a_1$ | | | |
| MD irrelevant for ...... | MD 32510: FRICT_COMP_ADAPT_ENABLE = 0<br>MD 32490: FRICT_COMP_MODE = 2 (neural QEC) | | | |
| Related to .... | MD 32500: FRICT_COMP_ENABLE Friction compensation active<br>MD 32510: FRICT_COMP_ADAPT_ENABLE Friction compensation adaptation active<br>MD 32520: FRICT_COMP_CONST_MAX Maximum friction compensation value<br>MD 32530: FRICT_COMP_CONST_MIN Minimum friction compensation value<br>MD 32560: FRICT_COMP_ACCEL2 Adaptation acceleration value 2<br>MD 32570: FRICT_COMP_ACCEL3 Adaptation acceleration value 3<br>MD 32540: FRICT_COMP_TIME Friction compensation time constant | | | |

| | | | | |
|---|---|---|---|---|
| **32560**<br>MD number | **FRICT_COMP_ACCEL2 [n]**<br>Adaptation acceleration value 2 [setpoint branch]: 0 | | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after NEW_CONF | | Protection level: 2 | | Unit: $m/s^2$ |
| Data type: DOUBLE | | | Applies from SW: 1.1 | |
| Meaning: | The adaptation acceleration value is only required if “Friction compensation with adaptation” is active.<br>The adaptation acceleration values 1 to 3 are interpolation points for defining the adaptation curve. The adaptation curve is subdivided into four ranges in which different friction compensation values apply.<br>Range B2 is defined by MD 32550: FRICT_COMP_ACCEL1 ($a_1$) and FRICT_COMP_ACCEL2 ($a_2$) (see Chapter 2, Subsection “Conv. friction compensation”).<br>For the injection amplitude within range B2 the following applies:<br>$\Delta n = \Delta n_{max}$ for $a_1 \leq a \leq a_2$ | | | |
| MD irrelevant for ...... | MD 32510: FRICT_COMP_ADAPT_ENABLE = 0<br>MD 32490: FRICT_COMP_MODE = 2 (neural QEC) | | | |
| Related to .... | MD 32500: FRICT_COMP_ENABLE Friction compensation active<br>MD 32510: FRICT_COMP_ADAPT_ENABLE Friction compensation adaptation active<br>MD 32520: FRICT_COMP_CONST_MAX Maximum friction compensation value<br>MD 32530: FRICT_COMP_CONST_MIN Minimum friction compensation value<br>MD 32550: FRICT_COMP_ACCEL1 Adaptation acceleration value 2<br>MD 32570: FRICT_COMP_ACCEL3 Adaptation acceleration value 3<br>MD 32540: FRICT_COMP_TIME Friction compensation time constant | | | |

| | | | |
|---|---|---|---|
| **32570**<br>MD number | **FRICT_COMP_ACCEL3 [n]**<br>Adaptation acceleration value 3 [setpoint branch]: 0 | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after NEW_CONF | Protection level: 2 | Unit: $m/s^2$ | |
| Data type: DOUBLE | Applies from SW: 1.1 | | |
| Meaning: | The adaptation acceleration value is only required if “Friction compensation with adaptation” is active.<br>The adaptation acceleration values 1 to 3 are interpolation points for defining the adaptation curve. The adaptation curve is subdivided into four ranges in which different friction compensation values apply.<br>Range B2 is defined by MD 32560: FRICT_COMP_ACCEL2 ($a_2$) and FRICT_COMP_ACCEL3 ($a_3$) (see Chapter 2, Subsection “Conv. friction compensation”).<br>For the injection amplitude within range B3 the following applies:<br><br>$\Delta n = \Delta n_{max} * \quad (1 - (a - a_2) / (a_3 - a_2))$ for $a_2 < a < a_3$<br><br>Range B4 applies to acceleration values > a3. The following applies to the injection amplitude within range B3:<br><br>$\Delta n = \Delta n_{min} \quad$ for $a \geq a_3$ | | |
| MD irrelevant for ...... | MD 32510: FRICT_COMP_ADAPT_ENABLE = 0<br>MD 32490: FRICT_COMP_MODE = 2 (neural QEC) | | |
| Related to .... | MD 32500: FRICT_COMP_ENABLE<br>MD 32510: FRICT_COMP_ADAPT_ENABLE<br>MD 32520: FRICT_COMP_CONST_MAX<br>MD 32530: FRICT_COMP_CONST_MIN<br>MD 32550: FRICT_COMP_ACCEL1<br>MD 32560: FRICT_COMP_ACCEL2<br>MD 32540: FRICT_COMP_TIME | Friction compensation active<br>Friction compensation adaptation active<br>Maximum friction compensation value<br>Minimum friction compensation value<br>Adaptation acceleration value 2<br>Adaptation acceleration value 2<br>Friction compensation time constant | |

| | | | |
|---|---|---|---|
| **32580**<br>MD number | **FRICT_COMP_INC_FACTOR**<br>Weighting factor of friction compensation value with short traversing movements | | |
| Default setting: 1 | Minimum input limit: 0 | Maximum input limit: 100 | |
| Changes effective after POWER ON | Protection level: 2 / 4 | Unit: % | |
| Data type: DOUBLE | Applies from SW: 2.1 | | |
| Meaning: | The optimum friction compensation value determined by the circularity test can cause overcompensation of this axis if compensation is activated and axial positioning movements are short.<br><br>In such cases, a better setting can be achieved by reducing the amplitude of the friction compensation value (conventional or quadrant error compensation with neural networks) and all positioning movements that are made within an interpolation cycle of the control.<br><br>The factor that has to be entered can be determined empirically and can be different from axis to axis because of the different friction conditions. The input range is between 0 to 100% of the value determined by the circularity test.<br><br>The default setting is 0; so that no compensation is performed for short traversing movements. | | |
| Related to .... | MD 32500: FRICT_COMP_ENABLE | Friction compensation active | |

<table>
<tr><td>32610<br>MD number</td><td colspan="5">VELO_FFW_WEIGHT[n]<br>Weighting factor for feedforward control</td></tr>
<tr><td colspan="2">Default setting: 1</td><td colspan="2">Minimum input limit: 0</td><td colspan="2">Maximum input limit: plus</td></tr>
<tr><td colspan="2">Changes effective after NEW_CONF</td><td colspan="2">Protection level: 2 / 7</td><td colspan="2">Unit: Factor</td></tr>
<tr><td colspan="3">Data type: DOUBLE</td><td colspan="3">Applies from SW: 1.1, changed from SW 5.1</td></tr>
<tr><td>Meaning:</td><td colspan="5">From SW 1.1 to SW 4.4 feedforward control factor for speed feedforward control<br>In the case of speed feedforward control, a velocity setpoint is also applied directly to the input of the speed controller (see Chapter 2, Section “Speed feedforward control”). This additional setpoint can be weighted with a factor (called feedforward control factor).<br>To ensure that the speed feedforward control is set correctly, the equivalent time constant of the speed control loop must be determined precisely and entered in MD 32810: EQUIV_SPEEDCTRL_TIME.<br>If the equivalent time constant of the speed control loop is defined exactly, the feedforward control factor has a value of approximately 1. In this case, the system deviation is roughly zero (check by looking at the service display in the operating area Diagnosis).<br>If the feedforward control factor 0 is entered, feedforward control is deactivated. However, because the calculations are performed anyway, feedforward control must be deactivated with MD: FFW_MODE = 0.<br><br>From SW 5.1 weighting factor for feedforward control<br>dependent on the parameter set [n=0..5]<br>This factor weights the setpoint velocity before it is used for feedforward control of the drive.<br><br>Default: MD 32610: VELO_FFW_WEIGHT = 1.0<br>This default weighting factor (1.0) acts as follows:<br>• On digital drives typically to ensure the setpoint speed is strictly adhered to.<br>• On analog drives the drive actuator gain errors can be compensated for, so that the actual speed is exactly equal to the setpoint speed (this reduces the following error when using feedforward control).<br><br>MD 32610: VELO_FFW_WEIGHT < 1.0<br>With both drives types this weighting factor allows the effect of the feedforward control to be gradually removed if the machine is traversing with a speed characteristic that is too stiff and other measures (e.g. jerk limitation) are not to be used. This reduces any overshooting, which results in increased error occurrence at curved contours, e.g. on circles.<br><br>Generally the settings for the feedforward control are better when the weighting factor remains set to (1.0), and overshooting is removed by means of jerk limitation and balancing filter MD 32810: EQUIV_SPEEDCTRL_TIME.<br><br>MD 32610: VELO_FFW_WEIGHT = 0.0<br>This setting provides position control only without feedforward control.<br><br>MD 32610: VELO_FFW_WEIGHT > 1.0<br>The contour monitor also takes weighting factors higher than 1 into account.<br>In individual cases, it may be necessary to increase the contour monitoring tolerance band in MD 36400: CONTOUR_TOL.</td></tr>
<tr><td>MD irrelevant for ......</td><td colspan="5">MD 32620: FFW_MODE = 0 or 2</td></tr>
<tr><td>Related to ....</td><td colspan="5">MD 32620: FFW_MODE<br>MD 32630: FFW_ACTIVATION_MODE<br>MD 32810: EQUIV_SPEEDCTRL_TIME</td></tr>
</table>

<table>
<tr><td>32620<br>MD number</td><td colspan="3">FFW_MODE<br>Feedforward control mode</td></tr>
<tr><td>Default setting: 1</td><td>Minimum input limit: 0</td><td colspan="2">Maximum input limit: 4</td></tr>
<tr><td colspan="2">Changes effective after RESET</td><td>Protection level: 2 / 7</td><td>Unit: –</td></tr>
<tr><td colspan="2">Data type: Byte</td><td colspan="2">Applies from SW: 4.3 extended in SW 5.1 and higher</td></tr>
<tr><td>Meaning:</td><td colspan="3">FFW_MODE defines the feedforward control mode to be applied on an axis-specific basis.<br>0 = No feedforward control<br>1 = Velocity feedforward control with PT1 balancing<br>2 = Torque feedforward control with PT1 balancing<br>(only possible with SINUMERIK 840D)<br><br>Extending the selection with values 3 and 4 with SW 5.1 and higher<br>for SINUMERIK 840D/810D only<br>The default is 1 for compatibility with earlier software versions.<br><br>The default for SINUMERIK 840Di is 3<br>3= Velocity feedforward control with Tt balancing<br>4= Torque feedforward control with Tt balancing<br>(only possible with SINUMERIK 840D)<br><br>Torque feedforward control is an option that must be enabled.<br><br>FFWON and FFWOF are used to activate and deactivate the feedforward control for specific channels on all axes.<br>To prevent these instructions from changing the feedforward control for individual axes, the setting can be activated or deactivated permanently in machine data MD 32630: FFW_ACTIVATION_MODE (see also MD 32630).<br><br>If a feedforward control mode is selected (speed or torque feedforward control), it can be programmed additionally in MD 32630: FFW_ACTIVATION_MODE whether the feedforward control can be activated or deactivated by the parts program.</td></tr>
<tr><td>Application example(s)</td><td colspan="3">In order to achieve excellent machining accuracy at high path velocities, contour inaccuracies due to following error can be eliminated using feedforward control.</td></tr>
<tr><td>Related to ....</td><td colspan="3">MD 32630: FFW_ACTIVATION_MODE<br>MD 32610: VELO_FFW_WEIGHT<br>MD 32650: AX_INERTIA</td></tr>
</table>

<table>
<tr><td>32630<br>MD number</td><td colspan="3">FFW_ACTIVATION_MODE<br>Activate feedforward control from program</td></tr>
<tr><td>Default setting: 1</td><td>Minimum input limit: 0</td><td colspan="2">Maximum input limit: 1</td></tr>
<tr><td colspan="2">Changes effective after RESET</td><td>Protection level: 0 / 0</td><td>Unit: –</td></tr>
<tr><td colspan="2">Data type: Byte</td><td colspan="2">Applies from SW: 4.3</td></tr>
<tr><td>Meaning:</td><td colspan="3">FFW_ACTIVATION_MODE can be set to define whether the feedforward control for this axis/spindle can be switched on and off in the parts program.<br>0 = The feedforward control cannot be switched on or off by high-level language elements FFWON or FFWOF.<br>The condition set in MD: FFW_MODE is always active for the axis/spindle.<br>1 = Feedforward control can be switched on/off with FFWON or FFWOF.<br>The default setting is programmed in channel-specific data MD 20150: GCODE_RESET_VALUES. This setting is valid even before the first NC block has been executed.<br>The last condition to be active remains active even after Reset (and therefore with JOG).<br>Because the feedforward control for all axes of a channel is switched on/off with FFWON or FFWOF, MD:<br>FFW_ACTIVATION_MODE should therefore have identical settings for axes that interpolate with each other.</td></tr>
<tr><td>Related to ....</td><td colspan="3">MD 32620: FFW_MODE<br>MD 20150: GCODE_RESET_VALUES</td></tr>
<tr><td>References</td><td colspan="3">/PA/, “Programming Guide Fundamentals”</td></tr>
</table>

| 32640<br>MD number | STIFFNESS_CONTROL_ENABLE<br>Activate dynamic stiffness control | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 |
| Changes effective after NEW_CONF | Protection level: 2 / 7 | Unit: – |
| Data type: | Applies from SW: 4.1 | |
| Meaning: | Activate dynamic stiffness control if bit is set.<br><br>With active dynamic stiffness control, higher servo gain factors are possible (MD 32200: POSCTRL_GAIN).<br><br>Due to the higher computing load in SIMODRIVE 611 digital , it may be necessary to adapt the settings of the sampling cycle (current/drive module cycle) in the 611D.<br>(This does not apply to PROFIBUS-DP drives)<br><br>For a single-axis drive module, the default setting (125 µs current, 125 µs speed controller cycle) is sufficient. The speed controller cycle might have to be increased (to 250 µs) for two-axis modules.<br><br>**Note:**<br>Up to SW 6.4, because of the implementation in the SIMODRIVE 611 digital drive, the stiffness control is only possible with the motor measuring system. | |
| Related to .... | MD 32642: STIFFNESS_CONTROL_CONFIG<br>MD 32644: STIFFNESS_DELAY_TIME | |
| References | /FBA/, DD2 Dynamic Stiffness Control (DSC) Subsection | |

| 32642<br>MD number | STIFFNESS_CONTROL_CONFIG<br>Config. dynamic stiffness control | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: BOOLEAN | Applies from SW: 6.3 | |
| Meaning: | Dynamic Stiffness Control Configuration (DSC):<br><br>Normal case: 0 = DSC in drive uses indirect measuring system<br>PROFIBUS-DP drives: 1 = DSC in drive uses direct measuring system<br><br>**Note:**<br>Up to SW 6.4, DSC with a direct measuring system is possible only on the PROFIBUS-DP with SIMODRIVE 611 universal drives. | |
| Related to .... | MD 32640: STIFFNESS_CONTROL_ENABLE<br>MD 32644: STIFFNESS_DELAY_TIME | |
| References | /FBA/, DD2 Dynamic Stiffness Control (DSC) Subsection | |

| 32644<br>MD number | STIFFNESS_DELAY_TIME<br>Dyn. stiffness control: Delay | |
|---|---|---|
| Default setting: 0.0 | Minimum input limit: –0.02 | Maximum input limit: 0.02 |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: s |
| Data type: DOUBLE | Applies from SW: 6.3 | |
| Meaning: | Effective only for PROFIBUS-DP drives<br>Configuration of a compensation deadtime for Dynamic Stiffness Control (DSC) with optimized DP cycle (e.g. SIMODRIVE 611 universal).<br>With machine data MD 32644: STIFFNESS_DELAY_TIME, fine adjustment for the DSC function is also possible for third-party PROFIBUS-DP drives.<br><br>On SIMODRIVE 611 digital drives, the complete adjustment is performed automatically within the control. Readjustment is not necessary. | |
| Related to .... | MD 32640: STIFFNESS_CONTROL_ENABLE<br>MD 32642: STIFFNESS_CONTROL_CONFIG | |
| References | /FBA/, DD2 Dynamic Stiffness Control (DSC) Subsection | |

| | | | | |
|---|---|---|---|---|
| **32650**<br>MD number | **AX_INERTIA**<br>Moment of inertia for torque feedforward control | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: plus |
| Changes effective after NEW_CONF | | Protection level: 2 | | Unit: $kgm^2$ |
| Data type: DOUBLE | | | Applies from SW: 1.1 | |
| Meaning: | In the case of torque feedforward control, an additional current setpoint proportional to the torque is applied directly to the current controller input (see Chapter 2, Subsection “Torque feedforward control”). This value is formed using the acceleration and the moment of inertia. The equivalent time constant of the current control loop must be defined for this purpose and entered in MD 32800: EQUIV_CURRCTRL_TIME.<br>The total moment of inertia of the axis (drive + load) must also be entered in AX_INERTIA (total moment of inertia referred to motor shaft according to data supplied by machine manufacturer).<br>When AX_INERTIA and MD 32800: EQUIV_CURRCTRL_ TIME are set correctly, the following error is almost zero even during acceleration (check this by looking at the following error in the service display).<br>The torque feedforward control is deactivated if AX_INERTIA is set to 0. However, because the calculations are performed anyway, torque feedforward control must always be deactivated with MD: FFW_MODE = 0 or 1.<br>Because of the direct current setpoint injection, torque feedforward control is only possible on digital drives (SIMODRIVE 611D). | | | |
| MD irrelevant for ... ... | SINUMERIK MD 32620: FFW_MODE = 0 or 1 | | | |
| Application example(s) | Torque feedforward control is required to achieve high contour accuracy where the demands on the dynamics are great. | | | |
| Related to .... | MD 32620: FFW_MODE<br>MD 32630: FFW_ACTIVATION_MODE<br>MD 32800: EQUIV_CURRCTRL_TIME | | | |

| | | | | |
|---|---|---|---|---|
| **32652**<br>MD number | **AX_MASS**<br>Axis mass for torque feedforward control | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: plus |
| Changes effective after NEW_CONF | | Protection level: 2 / 7 | | Unit: kg |
| Data type: DOUBLE | | | Applies from SW: 4.1 | |
| Meaning: | Mass of axis for torque feedforward control.<br>This MD is used instead of AX_INERTIA on linear drives (DRIVE_TYPE=3). | | | |
| Related to .... | | | | |
| References | | | | |

| | | | | |
|---|---|---|---|---|
| **32700**<br>MD number | **ENC_COMP_ENABLE[n]**<br>LEC active [n] | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 |
| Changes effective after NEW_CONF | | Protection level: 2 / 7 | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 1.1 | |
| Meaning: | 1: ’LEC’ is activated for the axis/measuring system.<br>With LEC, leadscrew errors and measuring system errors can be compensated.<br>The function is only enabled internally if the measuring system has been referenced (IS: “Referenced/synchronized” = 1).<br>Write protection function (compensation values) active.<br>0: Interpolatory compensation is not active for the axis/measuring system.<br><br>Index [n] has the following coding:<br>[Encoder no.]: 0 or 1 | | | |
| Related to .... | MD: MM_ENC_COMP_MAX_POINTS Number of interpolation points for LEC<br>IS “Referenced/synchronized 1”<br>IS “Referenced/synchronized 2” | | | |

| | | | |
|---|---|---|---|
| **32710**<br>MD number | **CEC_ENABLE**<br>Enabling of beam sag compensation | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after NEW_CONF | Protection level: 2 / 4 | Unit: – | |
| Data type: BOOLEAN | Applies from SW: 2.0 | | |
| Meaning: | 1: 'Beam sag compensation' is enabled for the compensation axis.<br>With "beam sag compensation", inter-axis geometry errors (e.g. beam sag and angularity errors) can be compensated. The function is not enabled in the control until the following conditions have been fulfilled:<br>• Option 'Interpolatory compensation' is set<br>• Associated compensation tables are available<br>• Evaluation of the required compensation table is enabled (SD: CEC_TABLE_ENABLE[t] = 1)<br>• The position measuring system required is referenced (IS: "Referenced/synchronized" = 1).<br>• Write protection function (compensation values) active.<br><br>0: 'Beam sag compensation' is not enabled for the compensation axis. | | |
| Related to .... | MD: MM_CEC_MAX_POINTS[t]<br><br>SD: CEC_TABLE_ENABLE[t]<br><br>"Referenced/synchronized 1 or 2" | Number of interpolation points for beam sag compensation<br>Enable evaluation of beam sag com pensation table t IS<br>DB31-48, DBX60.4 or 60.5 | |

| 32711<br>MD number | CEC_SCALING_SYSTEM_METRIC<br>Measuring system of beam sag compensation | | | |
|---|---|---|---|---|
| Default setting: 1 | | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after RESET | | Protection level: 2 / 7 | | Unit: – |
| Data type: BOOLEAN | | Applies from SW: 5 | | |
| Meaning: | Compensation data are contained in:<br>• MD 32711=0: inch system<br>• MD 32711=1: metric system<br><br>The measuring system can be configured for all beam sag compensation tables that affect the same axis.<br>Hereby all position entries are interpolated together with the calculated total axial compensation value in the configured measuring system.<br>External table conversions after the measuring system has been switched over are no longer necessary.<br>Axial configuration of the measuring system is necessary, as only the total axial compensation value is referring unambiguously to a position, not the individual table contents that are calculated in relation to one another.<br><br>**Note:** Only effective when MD 10260: CONVERT_SCALING_SYSTEM=1. (see /G2/) | | | |
| Related to .... | MD 10260: CONVERT_SCALING_SYSTEM | Basic system switchover active | | |

| MD number | System of measurement of sag compensation | | | |
|---|---|---|---|---|
| 32720<br>MD number | CEC_MAX_SUM<br>Maximum compensation value for beam sag compensation | | | |
| Default setting: 1 | | Minimum input limit: 0 | Maximum input limit: 10 | |
| Changes effective after NEW_CONF | | Protection level: 2 / 4 | | Unit:<br>Linear axis: mm<br>Rotary axis: Degrees |
| Data type: DOUBLE | | Applies from SW: 2.1 | | |
| Meaning: | In beam sag compensation, the absolute magnitude of the total compensation value (sum of compensation values of all compensation relations) is monitored axially with machine data value CEC_MAX_SUM.<br><br>If the determined total compensation value is larger than the maximum value, alarm 20124 is triggered. Program processing is not interrupted. The compensation value output as the additional setpoint is limited to the maximum value. | | | |
| MD irrelevant for ...... | Leadscrew error compensation<br>Backlash compensation<br>Temperature compensation | | | |
| Related to .... | MD: CEC_ ENABLE<br>SD: CEC_TABLE_ENABLE[t]<br><br>“Referenced/synchronized 1 or 2” | Enable beam sag compensation<br>Enable evaluation of beam sag com pensation table t IS<br>DB31-48, DBX60.4 or 60.5 | | |

| | | | |
|---|---|---|---|
| **32730**<br>MD number | **CEC_MAX_VELO**<br>Maximum permissible change value for beam sag compensation | | |
| Default setting: 10 | Minimum input limit: 0 | Maximum input limit: 100 | |
| Changes effective after NEW_CONF | Protection level: 2 / 7 | Unit: % | |
| Data type: DOUBLE | Applies from SW: 2.1 | | |
| Meaning: | In beam sag compensation, modification of the total compensation value (sum of the compensation values of all active compensation relations) is limited axially. The maximum change value is defined in this machine data as a percentage of MD 32000: MAX_AX_VELO (maximum axis velocity).<br>If the change in the total compensation value is greater than the maximum value, alarm 20125 is output. Program processing is however continued. The path not covered because of the limitation is made up as soon as the compensation value is no longer subject to limitation. | | |
| MD irrelevant for ... .. | Leadscrew error compensation<br>Backlash compensation<br>Temperature compensation | | |
| Related to .... | MD: CEC_ ENABLE<br>MD: MAX_AX_VELO<br>SD: CEC_TABLE_ENABLE[t]<br><br>"Referenced/synchronized 1 or 2" | Enable beam sag compensation<br>Maximum axis velocity<br>Enable evaluation of beam sag com pensation table t IS<br>DB31-48, DBX60.4 or 60.5 | |

| | | | |
|---|---|---|---|
| **32750**<br>MD number | **TEMP_COMP_TYPE**<br>Temperature compensation type | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 7 | |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: Hex | |
| Data type: BYTE | Applies from SW: 1.1, bit 2, in SW 6.1 and higher extended | | |
| Meaning: | The temperature compensation type effective for the machine axis is activated with machine data TEMP_COMP_TYPE.<br>A distinction is made between the following types:<br>TEMP_COMP_TYPE<br>Value = 0: No temperature compensation active<br>Bit 0 = 1: Positions **independent** temperature compensation active (compensation value with SD 43900: TEMP_COMP_ABS_VALUE)<br>Bit 1 = 1: Position **dependent** temperature compensation active (compensation value with SD 43910: TEMP_COMP_SLOPE and SD 43920: TEMP_COMP_REF_POSITION)<br>Bit 2 = 1 : Compensation active **in tool direction** (in order to activate temperature compensation in tool direction, machine data MD 20390: TOOL_TEMP_COMP_ON must also be set.)<br><br>Several compensation types can be activated at the same time.<br><br>The compensation values are overwritten via the PLC or synchronized actions.<br>To prevent velocity step changes on individual axes, the applied compensation values are limited via MD 32760: COMP_ADD_VELO_FACTOR.<br>Temperature compensation is an option that must be enabled. | | |
| Related to .... | SD 43900: TEMP_COMP_ABS_VALUE<br><br>SD 43920: TEMP_COMP_REF_POSITION<br><br>SD 43910: TEMP_COMP_SLOPE<br><br>MD 32760: COMP_ADD_VELO_FACTOR<br><br>MD 20390: TOOL_TEMP_COMP_ON | Position-dependent temperature compensation value<br>Reference position for position-dependent temperature compensation<br>Gradient for position-dependent temperature compensation<br>Velocity violation due to compensation<br>Activation of temperature compensation for tool length | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **32760**<br>MD number | **COMP_ADD_VELO_FACTOR**<br>Velocity violation due to compensation | | | | | |
| Default setting: 0,01 | | Minimum input limit: 0 | | Maximum input limit: 0,1 | | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: Factor | |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | | |
| Meaning: | With axial MD: COMP_ADD_VELO_FACTOR the maximum distance that can be traversed because of temperature compensation in one IPO cycle is limited.<br>If the resulting temperature compensation value is above this maximum, it is traversed over several IPO cycles. There is no alarm.<br>The maximum compensation value per IPO cycle is input as a factor with reference to the maximum axis velocity (MD: MAX_AX_VELO).<br>With this machine data the maximum gradient of the temperature compensation $\tan\beta_{max}$ is also limited.<br><br><u>Example of calculation of the maximum gradient $\tan\beta_{max}$:</u><br>1. Calculation of the interpolator closed-loop control (see Description of Functions Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control (G2))<br>Interpolator closed-loop control = basic system clock rate * factor for interpolation cycle<br>Interpolator closed-loop control = MD: SYSCLOCK_CYCLE_TIME * MD: IPO_SYSCLOCK_TIME_RATIO<br>Example: MD: SYSCLOCK_CYCLE_TIME = 0.004 [s]<br>MD: IPO_SYSCLOCK_TIME_RATIO = 3<br>⇒ Interpolator closed-loop control = 0.004 * 3 = 0.012 [s]<br>2. Calculation of the maximum velocity increase because of a change made to the temperature compensation parameter $\Delta v_{Tmax}$<br>$\Delta v_{Tmax}$ = MD: MAX_AX_VELO * MD: COMP_ADD_VELO_FACTOR<br>Example: MD: MAX_AX_VELO = 10 000 [mm/min ]<br>MD: COMP_ADD_VELO_FACTOR = 0.01<br>⇒ $\Delta v_{Tmax}$ = 10 000 * 0.01 = 100 [mm/min]<br>3. Calculation of the traverse distances per interpolator cycle<br>$S_1 \text{ (at } v_{max}) = 10\,000 \times \frac{0.012}{60} = 2.0 \text{ [mm]}$<br>$S_T \text{ (at } \Delta v_{Tmax}) = 100 \times \frac{0.012}{60} = 0.02 \text{ [mm]}$<br>4. Calculation of $\tan\beta_{max}$<br>$\tan\beta_{max} = \frac{S_T}{S_1} = \frac{0.02}{2} = 0.01$ (corresponds to the value of COMP_ADD_VELO_FACTOR)<br>⇒ $\beta_{max}$ = arc tan 0.01 = 0.57 degrees<br>With larger values of SD: TEMP_COMP_SLOPE the maximum gradient (here 0.57 degrees) for the position-dependent temperature compensation value is used internally. There is no alarm.<br><br>**Note:** Any additional velocity violation caused by temperature compensation must be taken into account when defining the limit value for velocity monitoring (MD: AX_VELO_LIMIT). | | | | | |
| MD irrelevant for ...... | TEMP_COMP_TYPE = 0, sag compensation, LEC, backlash compensation | | | | | |
| Related to .... | MD: TEMP_COMP_TYPE<br>SD: TEMP_COMP_ABS_VALUE<br>SD: TEMP_COMP_SLOPE<br><br>MD: MAX_AX_VELO<br>MD: AX_VELO_LIMIT<br>MD: IPO_SYSCLOCK_TIME_RATIO<br>MD: SYSCLOCK_CYCLE_TIME | | | | Temperature compensation type<br>Position-independent temperature compensation<br>Gradient for position-dependent temperature compensation<br>Maximum axis velocity<br>Limit value for velocity monitoring<br>Ratio basic system clock rate to IPO cycle<br>System clock cycle | |

| **32800**<br>MD number | **EQUIV_CURRCTRL_TIME [n]**<br>Equivalent time constant of current control loop | |
|---|---|---|
| Default setting: 0.0005 | Minimum input limit: 0 | Maximum input limit: plus |
| Changes effective after NEW_CONF | Protection level: 2 / 7 | Unit: s |
| Data type: DOUBLE | Applies from SW: 1.1 | |
| Meaning: | This time constant must equal the equivalent time constant of the closed current control loop.<br>It is used for parameterization of the torque feedforward control and for calculation of the dynamic following error model (contour monitoring).<br>In order to set the torque feedforward control correctly, the equivalent time constant of the current control loop must be determined precisely by measuring the step response of the current control loop.<br>With SIMODRIVE 611D the settling process can be displayed using the installation tool.<br>Index[n] has the following coding:<br>[control parameter block number]: 0 to 5<br>(**References:** /FB/, G2, "Velocities, Setpoint/Actual Value Systems, Cycle Times") | |
| Related to .... | MD: FFW_MODE Feedforward control type<br>MD: AX_INERTION Moment of inertia for speed feedforward control<br>MD: CONTOUR_TOL Tolerance band contour monitoring | |
| References | /IAD/ Installation and Start-Up Guide<br>/IAF/ Installation and Start-Up Guide | |

| **32810**<br>MD number | **EQUIV_SPEEDCTRL_TIME [n]**<br>Equivalent time constant of speed control loop | |
|---|---|---|
| Default setting: 0.004 | Minimum input limit: 0 | Maximum input limit: plus |
| Changes effective after NEW_CONF | Protection level: 2 / 7 | Unit: s |
| Data type: DOUBLE | Applies from SW: 1.1 | |
| Meaning: | This time constant must equal the equivalent time constant of the closed speed control loop.<br>It is used for parameterization of the symmetrization filter for the speed feedforward control and for calculation of the dynamic following error model (contour monitoring).<br>In order to set the speed feedforward control correctly, the equivalent time constant of the speed control loop must be determined precisely by measuring the step response of the speed control loop.<br>With SIMODRIVE 611D the settling process can be displayed using the installation tool.<br>Index[n] has the following coding:<br>[control parameter block number]: 0 to 5<br>**(References:** /FB/, G2, "Velocities, Setpoint/Actual Value Systems, Cycle Times") | |
| Related to .... | MD: FFW_MODE Feedforward control type<br>MD: VELO_FFW_WEIGHT Moment of inertia for speed feedforward control<br>MD: CONTOUR_TOL Tolerance band contour monitoring | |
| References | /IAF/, Installation and Start-Up Guide<br>/IAD/, Installation and Start-Up Guide | |

| | | | |
|---|---|---|---|
| **36500**<br>MD number | **ENC_CHANGE_TOL**<br>Maximum tolerance for position actual value switchover | | |
| Default setting: 0.1 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after NEW_CONF | Protection level: 2 / 7 | Unit: mm, degrees | |
| Data type: DOUBLE | Applies from SW: 1.1 | | |
| Meaning: | The permissible deviation between the actual values of the two measuring systems is entered in the MD.<br>This tolerance must not be violated when switching from one measuring system to the other for closed-loop control in order to avoid too large compensatory processes. Otherwise error message 25100, “Axis %1 Measuring system switchover not possible” is generated and switchover between the two systems does not take place.<br>**SW 5.3 and higher**<br>This MD is used to manage large backlash compensation values. It ensures that the backlash is not switched through to the actual value all at once, but in n steps with an increment size as set in MD 36500: ENC_CHANGE_TOL. Inclusion of the backlash thus takes n servo cycles.<br>If the time elapsed until the full inclusion of the backlash is excessive, zero speed monitoring alarms may be triggered. The original method of injecting the backlash compensation value is used if<br>MD 36500: ENC_CHANGE_TOL is set higher than<br>MD 32450: BACKLASH. | | |
| MD irrelevant for ...... | This MD is irrelevant for MD 30200: NUM_ENCS = 0 or 1. | | |
| Application example(s) | To avoid too large compensatory processes when switching measuring systems. | | |
| Related to .... | MD 32450: BACKLASH | | |

| | | | |
|---|---|---|---|
| **36700**<br>MD number | **DRIFT_ENABLE**<br>Automatic drift compensation | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after NEW_CONF | Protection level: 840D:0 | Unit: – | |
| Data type: BOOLEAN | Applies from SW: 1.1 | | |
| Meaning: | MD: DRIFT_ENABLE activates automatic drift compensation.<br>1: Automatic drift compensation is active (only for position-controlled axes/spindles).<br>With automatic drift compensation at zero speed, the control constantly determines the additional drift value required so that the value zero is reached for the following error (compensation criterion).<br>The total drift value is therefore composed of the basic drift value (MD: DRIFT_VALUE) and the additional drift value (see Fig. 2-21).<br>0: Automatic drift compensation is not active.<br>The drift value is only formed from the basic drift value (MD: DRIFT_VALUE). | | |
| MD irrelevant for ...... | 840D or for axes/spindles which are not position-controlled | | |
| Related to .... | MD: DRIFT_LIMIT<br>MD: DRIFT_VALUE | Drift limit value for automatic drift compensation<br>Drift basic value | |

| | | | |
|---|---|---|---|
| **36710**<br>MD number | **DRIFT_LIMIT**<br>Drift limit value for automatic drift compensation | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after NEW_CONF | Protection level:<br>840D: 0<br>FM-NC: 2 | Unit:<br>% of manipulated variable (e.g. 10 V ≙ 100%) | |
| Data type: DOUBLE | Applies from SW: 1.1 | | |
| Meaning: | MD: DRIFT_LIMIT, the magnitude of the additional drift value determined during automatic drift compensation can be limited.<br>If the additional drift value exceeds the value entered in MD: DRIFT_LIMIT, alarm 25070 “Drift value too large” is signaled and the additional drift value is limited to this value. | | |
| MD irrelevant for ...... | SINUMERIK 840D or<br>MD: DRIFT_ENABLE = 0 | | |
| Related to .... | MD: DRIFT_ENABLE | Automatic drift compensation | |

| | | | | | |
|---|---|---|---|---|---|
| **36720**<br>MD number | **DRIFT_VALUE**<br>Drift basic value | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: | |
| Changes effective after NEW_CONF | | | Protection level: 840D:0 | | Unit: % |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | |
| Meaning: | The basic drift value entered in MD: DRIFT_VALUE is always injected as an additional speed setpoint.<br>The basic drift value is **always** active (independently of the MD: DRIFT_ENABLE)!<br>While the automatic drift compensation only applies to position-controlled axes, the basic drift value is also active for speed-controlled axes/spindles.<br>**Note:** Digital drives have no drift!<br>For drives on the Profibus:<br>This MD can still however be used for “simple” drives on the Profibus which are experiencing drift problems. To avoid setting errors, however, this drift compensation on the Profibus is operative only if $MA_RATED_OUTVAL != 0 (i.e. the MD is inactive when the interfaces between the NC and drive have been automatically adjusted).<br>Normalization: The input value refers to the interface normalization according to $MA_RATED_OUTVAL, $MA_RATED_VELO and $MA_CTRLOUT_LIMIT.<br><br>Note: When the DSC ($MA_STIFFNESS_CONTROL_ENABLE=1) function is used, drift compensation must not be active on the Profibus as unpredictable speed fluctuations will otherwise occur when DSC is activated or deactivated. | | | | |
| MD irrelevant for ...... | SINUMERIK 840D | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **38000**<br>MD number | **MM_ENC_COMP_MAX_POINTS[n]**<br>Number of interpolation points for LEC (SRAM) | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 5000 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | For leadscrew error compensation, the number of interpolation points required per measuring system must be defined.<br>The required number can be calculated as follows using the defined parameters (see Subsection 2.3.2)<br>MD: MM_ENC_COMP_MAX_POINTS = $\frac{\$AA\_ENC\_COMP\_MAX - \$AA\_ENC\_COMP\_MIN}{\$AA\_ENC\_COMP\_STEP} + 1$<br><br>$AA_ENC_COMP_MIN Initial position (system variable)<br>$AA_ENC_COMP_MAX End position (system variable)<br>$AA_ENC_COMP_STEP Distance betw. interpolation points (system variable)<br><br>In selecting the number of interpolation points and the distance between them, it is important to take account of the resulting size of the compensation table and the required space in the backed-up NC user memory (SRAM). 8 bytes are required per compensation value (interpolation point).<br>Index[n] has the following coding: [encoder no.]: 0 or 1 | | | | |
| Special cases, errors, ...... | **Notice:**<br>After any change in MD: MM_ENC_COMP_MAX_POINTS, the backed-up NC user memory is reallocated automatically on power up.<br>All data in the backed-up NC user memory are then lost (e.g. parts programs, tool offsets etc.). Alarm 6020 “Machine data changed – memory reallocated” is signaled.<br>If reallocation of the NC user memory fails because the total memory capacity available is not sufficient, alarm 6000 “Memory allocation made with standard machine data” is signaled.<br>In this case the NC user memory is allocated using the default values of the standard machine data.<br>**References:** /FB/, S7, “Memory Configuration”<br>/DA/, “Diagnostic Guide” | | | | |
| Related to .... | MD: ENC_ COMP_ ENABLE[n] Interpolatory compensation active | | | | |
| References | /FB/, S7, “Memory Configuration” | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **38010**<br>MD number | **MM_QEC_MAX_POINTS**<br>Maximum number of compensation values for QEC with neural networks | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1024 | |
| Changes effective after POWER ON | | | Protection level: 2 / 4 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 2.1 | | |
| Meaning: | In quadrant error compensation with neural networks (QEC) the number of required compensation values must be entered for every axis to be compensated.<br><br>The required number can be calculated as follows using the defined parameters (see Subsection 2.6.2):<br><br>MM_QEC_MAX_POINTS $\geq$ ($AA_QEC_COARSE_STEPS + 1) *<br>$AA_QEC_FINE_STEPS<br><br>$AA_QEC_COARSE_STEPS Coarse quantiz. of characteristic (system variable)<br>$AA_QEC_FINE_STEPS Fine quantization of characteristic (system variable)<br><br>For “direction-dependent” compensation the number must be greater than or equal to double value of this product.<br><br>When selecting coarse or fine quantization, the resulting size of the compensation table and the memory required for it in the non-volatile user memory must be taken into account. 4 bytes are required for every compensation value. If the value 0 is entered, no memory is reserved for the table; i.e. the table does not exist and the function can therefore not be activated. | | | | | |
| Special cases, errors, ...... | **Caution!**<br>If MD: MM_QEC_MAX_POINTS is altered, the non-volatile user memory is automatically re-allocated on system power-on. This deletes all the user data in the non-volatile user memory (e.g. drive and MMC machine data, tool offsets, parts programs etc.).<br><br>**Note:**<br>Because the exact number of required interpolation points is not exactly known during the first installation of the function, a large number should be chosen initially. As soon as the characteristics are recorded and saved, the number can be reduced to the required size. After performing a power ON again, the saved characteristics can be reloaded. | | | | |
| References | /FB/, S7, “Memory Configuration” | | | | |

## 4.2 Description of setting data

<table>
<tr><td>41300<br>MD number</td><td colspan="4">CEC_TABLE_ENABLE[t]<br>Enable evaluation of beam sag compensation table [t]</td></tr>
<tr><td>Default setting: 0</td><td colspan="2">Minimum input limit: 0</td><td colspan="2">Maximum input limit: 1</td></tr>
<tr><td colspan="2">Changes effective immediately</td><td colspan="2">Protection level: 7</td><td>Unit: –</td></tr>
<tr><td colspan="3">Data type: BOOLEAN</td><td colspan="2">Applies from SW: 2.1</td></tr>
<tr><td>Meaning:</td><td colspan="4">1: Evaluation of compensation table [t] is enabled. The compensation table defines, for example, the compensation relation (assignment of basis to compensation axis) with [t] = index of compensation table (see MD: MM_CEC_MAX_POINTS).<br>In 'beam sag compensation' the compensation axis can be influenced by several tables. SD: CEC_TABLE_ENABLE[t] can be altered by the NC parts program or PLC user program to adapt the total compensation value of the machining application (e.g. switch over tables).<br>The compensation is not enabled in the control until the following conditions have been fulfilled:<br>• Option 'Interpolatory compensation' is set<br>• Assigned compensation tables exist<br>• Beam sag compensation for compensation axis is activated (MD: CEC_ENABLE = 1)<br>• The position measuring system required is referenced (IS: “Referenced/synchronized” = 1).<br>0: Evaluation of the beam sag compensation table [t] is not enabled.</td></tr>
<tr><td>Related to ....</td><td colspan="2">MD: MM_CEC_MAX_POINTS[t]<br>SD: CEC_TABLE_ENABLE[t]<br>IS “Referenced/synchronized 1”<br>IS “Referenced/synchronized 2”</td><td colspan="2">Number of interpolation points for beam sag compensation<br>Enable evaluation of beam sag compensation table t<br>DB31-48, DBX60.4<br>DB31-48, DBX60.50</td></tr>
</table>

<table>
<tr><td>41310<br>MD number</td><td colspan="4">CEC_TABLE_WEIGHT[t]<br>Weighting factor for beam sag compensation table [t]</td></tr>
<tr><td>Default setting: 1,0</td><td colspan="2">Minimum input limit: ***</td><td colspan="2">Maximum input limit: ***</td></tr>
<tr><td colspan="2">Changes effective immediately</td><td colspan="2">Protection level: 7</td><td>Unit: (factor)</td></tr>
<tr><td colspan="3">Data type: DOUBLE</td><td colspan="2">Applies from SW: 2.1</td></tr>
<tr><td>Meaning:</td><td colspan="4">The compensation value stored in the table [t] is multiplied by the weighting factor. When choosing the weighting factor, ensure that the resulting compensation value does not exceed the maximum value (MD: CEC_MAX_SUM).<br>With [t] = index of compensation table (see MD: MM_CEC_MAX_POINTS)<br>If, for example, the weight of the tools on the machine or workpiece to be machined differ greatly and affect the error curve by a change in amplitude, this can be corrected by changing the weighting factor. In beam sag compensation the weighting factor of the table can be altered for specific tools or workpieces by the PLC user program or the NC program by overwriting the setting data.<br>If, however, the progression of the characteristic curve is greatly changed because of differing weights, different compensation tables must be used.</td></tr>
<tr><td>Related to ....</td><td colspan="2">SD: CEC_TABLE_ENABLE[t]<br>MD: CEC_MAX_SUM</td><td colspan="2">Enable evaluation of beam sag compensation table t<br>Maximum compensation value for beam sag compensation</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| **43900**<br>SD number | **TEMP_COMP_ABS_VALUE**<br>Position-independent temperature compensation value | | | |
| Default setting: 0 | Minimum input limit: *** | | Maximum input limit: *** | |
| Changes effective after: immediately | | Protection level: MMCMD 9220 | | Unit:<br>mm or degrees |
| Data type: DOUBLE | | | Applies from SW: 1.1 | |
| Meaning: | SD: TEMP_COMP_ABS_VALUE defines the **position-independent** temperature compensation value (see Fig. 2-2).<br>This value depends on the current temperature from the PLC (user program).<br>As soon as position-independent temperature compensation has been activated (MD: TEMP_COMP_TYPE = 1 or 3), this additional compensation value is traversed by the machine axis. | | | |
| SD irrelevant for ...... | MD: TEMP_COMP_TYPE = 0 or 2 | | | |
| Related to .... | MD: TEMP_COMP_TYPE<br>MD: COMP_ADD_VELO_FACTOR | | Temperature compensation type<br>Velocity violation due to compensation | |

| | | | | |
|---|---|---|---|---|
| **43910**<br>SD number | **TEMP_COMP_SLOPE**<br>Gradient for position-dependent temperature compensation | | | |
| Default setting: 0 | Minimum input limit: *** | | Maximum input limit: *** | |
| Changes effective after: immediately | | Protection level: MMCMD 9220 | | Unit:<br>mm or degrees |
| Data type: DOUBLE | | | Applies from SW: 1.1 | |
| Meaning: | With **position-dependent** temperature compensation, the error curve of the temperature-dependent actual-value deviation can often be approximated by a straight line. This straight line is defined by a reference point $P_0$ and a gradient $\tan\beta$ (see Fig. 2-2).<br>SD: TEMP_COMP_SLOPE defines the gradient $\tan\beta$. This gradient can be changed by the PLC user program as a function of the current temperature.<br>As soon as position-dependent temperature compensation is active (MD: TEMP_COMP_TYPE = 2 or 3), the axis traverses the compensation value calculated for the current actual position.<br>MD: COMP_ADD_VELO_FACTOR limits the maximum gradient $\tan\beta_{max}$ of the error curve. This maximum gradient cannot be exceeded. | | | |
| SD irrelevant for ...... | MD: TEMP_COMP_TYPE = 0 or 1 | | | |
| Special cases, errors, ...... | When TEMP_COMP_SLOPE is greater than $\tan\beta_{max}$, gradient $\tan\beta_{max}$ is used to calculate the position-dependent temperature compensation value internally. No alarm is output. | | | |
| Related to .... | MD: TEMP_COMP_TYPE<br>SD: TEMP_COMP_REF_POSITION<br><br>MD: COMP_ADD_VELO_FACTOR | | Temperature compensation type<br>Reference position for position-dependent temperature compensation<br>Velocity violation due to compensation | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **43920**<br>SD number | **TEMP_COMP_REF_POSITION**<br>Reference position for position-dependent temperature compensation | | | | | |
| Default setting: 0 | | Minimum input limit: *** | | | Maximum input limit: *** | |
| Changes effective after: immediately | | | Protection level: MMCMD 9220 | | | Unit:<br>mm or degrees |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | | |
| Meaning: | With **position-dependent** temperature compensation, the error curve of the temperature-dependent actual-value deviation can often be approximated by a straight line. This straight line is defined by a reference point $P_0$ and a gradient tanβ (see Fig. 2-2).<br>SD: TEMP_COMP_REF_POSITION defines the position of the reference point $P_0$. This reference point position can be changed by the PLC user program as a function of the current temperature.<br>As soon as position-dependent temperature compensation is active (MD: TEMP_COMP_TYPE = 2 or 3), the axis also traverses the compensation value calculated for the current actual position. | | | | | |
| SD irrelevant for ...... | MD: TEMP_COMP_TYPE = 0 or 1 | | | | | |
| Related to .... | MD: TEMP_COMP_TYPE<br>SD: TEMP_COMP_SLOPE | | | Temperature compensation type<br>Gradient for position-dependent temperature compensation | | |

■

# Notes

# Signal Descriptions 5

There are no separate signals for compensation.

■

# Example 6

– None –

■

# Data Fields, Lists 7

## 7.1 Interface signals

| DB number | Byte.Bit | Name | Refer-ence |
|---|---|---|---|
| General signals from NCK | | | |
| 10 | 108.7 | NC Ready | A2 |
| Mode groupspecific | | | |
| 11, ... | 6.3 | Mode group ready | |
| Channel-specific | | | |
| 21, ... | 36.5 | Channel ready | |
| Axis/spindle-specific | | | |
| 31, ... | 60.4 | Referenced/synchronized 1 | R1 |
| 31, ... | 60.5 | Referenced/synchronized 2 | R1 |
| 31, ... | 61.2 | Axis ready | |

## 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| General ($MN_...) | | | |
| 10050 | SYSCLOCK_CYCLE_TIME | System clock cycle | G2 |
| 10070 | IPO_SYSCLOCK_TIME_RATIO | Factor for interpolator cycle | G2 |
| 10082 | CTRLOUT_LEAD_TIME | Shift of setpoint transfer time | |
| 10083 | CTRLOUT_LEAD_TIME_MAX | Maximum permissible setting for shift of setpoint transfer time | |
| 10088 | REBOOT_DELAY_TIME | Reboot delay | |
| 18342 | MM_CEC_MAX_ POINTS[t] | Maximum number of interpolation points for the beam sag compensation | |
| Channel-specific ($MA_...) | | | |
| 20150 | GCODE_RESET_VALUES | Reset G groups | K1 |
| Axis-specific ($MC_ ... ) | | | |
| 32000 | MAX_AX_VELO | Maximum axis velocity | G2 |
| 32200 | POSCTRL_GAIN | Servo gain factor | G2 |
| 32450 | BACKLASH[n] | Backlash | |
| 32452 | BACKLASH_FACTOR[n] | Weighting factor for backlash | |
| 32460 | TORQUE_OFFSET | Additional torque for electr. weight compensation | |
| 32490 | FRICT_COMP_MODE | Type of friction compensation | |
| 32500 | FRICT_COMP_ENABLE | Friction compensation active | |
| 32510 | FRICT_COMP_ADAPT_ENABLE [n] | Friction compensation adaptation active | |
| 32520 | FRICT_COMP_CONST_MAX [n] | Maximum friction compensation value | |
| 32530 | FRICT_COMP_CONST_MIN [n] | Minimum friction compensation value | |
| 32540 | FRICT_COMP_TIME[n] | Friction compensation time constant | |
| 32550 | FRICT_COMP_ACCEL1 [n] | Adaptation acceleration value 1 | |
| 32560 | FRICT_COMP_ACCEL2 [n] | Adaptation acceleration value 2 | |
| 32570 | FRICT_COMP_ACCEL3 [n] | Adaptation acceleration value 3 | |
| 32580 | FRICT_COMP_INC_FACTOR | Weighting factor of friction compensation value with short traversing movements | |
| 32610 | VELO_FFW_WEIGHT | Feedforward control factor for speed feedforward control from SW 5.1 weighting factor for feedforward control | |
| 32620 | FFW_MODE | Feedforward control mode | |
| 32630 | FFW_ACTIVATION_MODE | Activate feedforward control from program | |
| 32640 | STIFFNESS_CONTROL_ENABLE | Activate dynamic stiffness control | |
| 32642 | STIFFNES_CONTROL_CONFIG | Config. dynamic stiffness control | |
| 32644 | STIFFNESS_DELAY_TIME | Dyn. stiffness control: Delay | |
| 32650 | AX_INERTIA | Moment of inertia for torque feedforward control | |
| 32652 | AX_MASS | Axis mass for torque feedforward control | |
| 32700 | ENC_COMP_ENABLE[n] | Interpolatory compensation active | |
| 32710 | CEC_ENABLE | Enabling of beam sag compensation | |
| 32711 | CEC_SCALING_SYSTEM_METRIC | System of measurement of sag compensation | |

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| 32720 | CEC_MAX_SUM | Maximum compensation value for beam sag compensation | |
| 32730 | CEC_MAX_VELO | Maximum value of change for beam sag compensation | |
| 32750 | TEMP_COMP_TYPE | Temperature compensation type | |
| 32760 | COMP_ADD_VELO_FACTOR | Velocity violation caused by compensation | |
| 32800 | EQUIV_CURRCTRL_TIME [n] | Equivalent time constant of current control loop | |
| 32810 | EQUIV_SPEEDCTRL_TIME [n] | Equivalent time constant of the speed control loop | |
| 36200 | AX_VELO_LIMIT | Limit value for velocity monitoring | A3 |
| 36400 | CONTOUR_TOL | Tolerance band contour monitoring | A3 |
| 36500 | ENC_CHANGE_TOL | Maximum tolerance for position actual value switchover | G2 |
| 36700 | DRIFT_ENABLE | Automatic drift compensation | |
| 36710 | DRIFT_LIMIT | Drift limit value for automatic drift compensation | |
| 36720 | DRIFT_VALUE | Drift basic value | |
| 38000 | MM_ENC_COMP_MAX_POINTS[n] | Number of intermediate points with interpola-tory compensation | |
| 38010 | MM_QEC_MAX_POINTS | Maximum number of compensation values for QEC with neural networks | |
| SIMODRIVE 611D machine data ($MD_...) | | | |
| 1004 | CTRL_CONFIG | Configuration structure | IAD |
| 1117 | MOTOR_INERTIA | Motor moment of inertia | IAD |

## 7.3 Setting data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| General ($MN_ ... ) | | | |
| 41300 | CEC_TABLE_ENABLE[t] | Enable evaluation of beam sag compensation table | |
| 41310 | CEC_TABLE_WEIGHT[t] | Weighting factor for beam sag compensation table | |
| Axis-specific ($SA_...) | | | |
| 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 | |

## 7.4 Alarms

For detailed descriptions of the alarms, please refer to
**References:** /DA/, "Diagnostics Guide"
and the online help of MMC 101/102/103 systems.

■

# SINUMERIK 840D/840Di/810D Description of Functions Extended Functions (FB2)

# Mode Groups, Channels, Axis Replacement (K5)

■

# 1 Brief Description

**Mode groups**

A mode group is a collection of machine axes, spindles and channels which are programmed to form a unit. A mode group can, in principle, be compared 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**

In the standard case a mode group exists and is described in

**References:** /FB/, K1, "Mode Group, Channels, Program Operation Mode"

---

**Channels**

Every channel has its own program decoding, block preparation and interpolation functions. A channel can process a part program independently.

---

**Note**

In the standard case a channel exists and is described in

**References:** /FB/, K1, "Mode Group, Channels, Program Operation Mode"

---

The processes in several channels of a mode group can be synchronized in the parts programs.

**Axis/spindle replacement**

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 replacement" it is possible to enable an axis/spindle and to allocate it to another channel, that means to replace the axis/spindle.

In SW 3 and higher, axis/spindle replacement can be activated both via the parts program and via the PLC program.

■

## Notes

# Detailed Description 2

## 2.1 Mode groups

**Mode groups**

A mode group combines NC channels with axes and spindles to form a machining unit.

A mode group contains all those channels that always have to operate in the same mode.

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 parts 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 MD 10010: 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:** /FB/, K1, “Mode Group, Channels, Program Operation Mode”

## 2.2 Channels

**Note**

A description of the terms Channel, Channel Configuration, Channel States, Effects of Commands/Signals, etc. for the first channel can be found in

**References:** /FB/, K1, “Mode Group, Channels, Program Operation Mode”

For all other channels, this information applies, too.

### 2.2.1 Channel synchronization (program coordination)

## General

**Definition**

As an example, double-slide machining operations or real-time processes can only be carried out if it is possible to synchronize processing in two channels. The channels affected shall perform certain processing procedures time-matched. To allow time-matched processing, the relevant channels must be joined to form a synchronization group (mode group).

The channel synchronization is programmed only via the NC language. The affected channels must be assigned to the **same mode group**.

**Coordination**

If several channels are involved in the machining of a workpiece, it may be necessary to synchronize program runs in the individual channels.
Special statements (commands) are provided for this program coordination. In each case, they are listed in one block.

Table 2-1 Program coordination instructions

| Instruction | Meaning |
|---|---|
| | **SW 3** |
| INIT(n,“identifier”,“q”) | Selection of a program for processing in a certain channel<br>Acknowledgement mode:<br>**n** (without) or<br>**s** (synchronous)<br>Name of program with path<br>Number of channel: Values 1 to 4 possible |
| CLEAR (identifier) | Deletion of a program indicating the program identifier |
| START (n,n,n,) | Start of the programs selected in other channels<br><br>Enumeration of channel numbers: Values 1 to 4 possible |
| WAITM (Mnr, n, n, n, n) | Wait for mark number Mnr for program synchronization in the specified channels n (channel used can be indicated, but this is optional). The mark number must be identical in all channels.<br>Numbers 0 to 9 can be selected. |
| WAITE (n,n,n) | Waiting for the program end of the channels indicated<br>(do not indicate program coordination channel) |
| | **SW 4** |
| SETM(Mnr1, Mnr2, ...Mnri) | Set wait marks Mnr1, Mnr2, ...Mnri for conditional wait with WAITMC() for the channel in which SETM() is issued. The channel thus declares to its partner channels that its wait characteristic is fulfilled.<br>The command can be activated in synchronized actions. Up to 10 marks (0–9) can be set using one command. |
| CLEARM(Mnr1, Mnr2, ...Mnri) | Delete wait marks Mnr1, Mnr2, ...Mnri for conditional wait with WAITMC() for the channel in which CLEARM() is issued. The channel thus declares to its partner channels that its wait characteristic is fulfilled.<br>The command can be activated in synchronized actions. Up to 10 marks (0 – 9) can be deleted using one command. |
| WAITMC(Mnr, n1, n2, ...) | Conditional wait in continuous-path mode for the specified wait characteristic Mnr from the specified channels n1, n2, ... nk. The program coordination channel can be indicated, 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. |

The number of marks depends on the CPU installed
CPU 572 —> 2 channels —> = 20
CPU 573 —> 10 channels —> =100

## SW 3

**Procedure**

When a WAITM() call is reached, the axes in the current channel are decelerated and wait until the mark number specified in the call arrives from the other channels to be synchronized. The group is synchronized when the other channels are also decelerated as they reach their WAITM() command. The synchronized channels then continue operation.

**Example of program coordination**

**Channel 1:**

```
%100
N10 INIT(2,"_N_200_MPF","n")
N11 START(2)
.                   ; Processing in channel 1

N80 WAITM(1,1,2)    ; Waiting for WAIT mark 1 in channel 1 and in channel 2
.                   ; Further processing in channel 1

N180 WAITM(2,1,2)   ; Waiting for WAIT mark 2 in channel 1 and in channel 2
.                   ; Further processing in channel 1

N200 WAITE(2)       ; Waiting for program end of channel 2
N201 M30            ; Program end channel 1, total end
.
.
```

**Channel 2:**

```
%200
.                   ; Processing in channel 2
N70 WAITM(1,1,2)    ; Waiting for WAIT mark 1 in channel 1 and in channel 2
.                   ; Further processing in channel 2
N270 WAITM(2,1,2)   ; Waiting for WAIT mark 2 in channel 1 and in channel 2
.                   ; Further processing in channel 2

N400 M30            ; Program end of channel 2
.
.
```

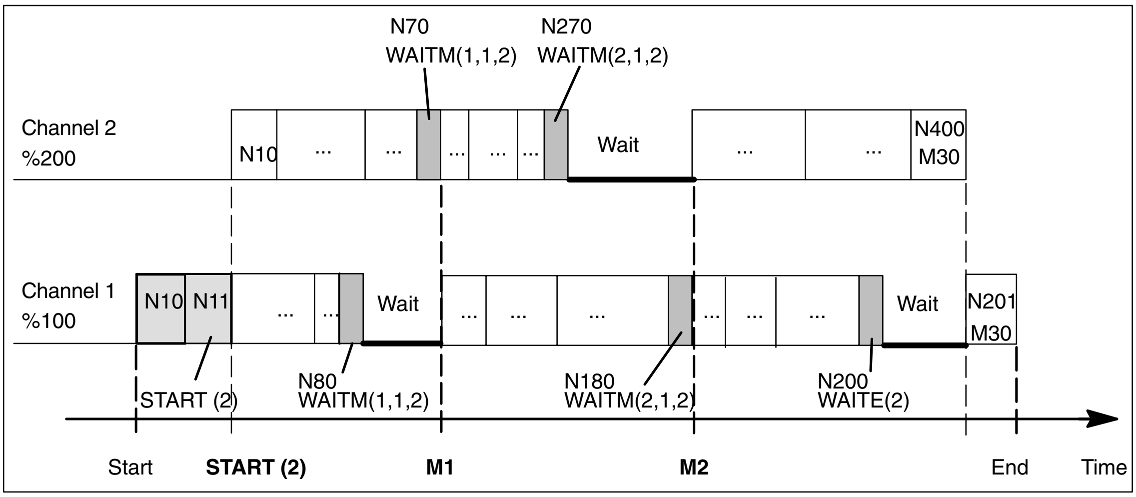



Fig. 2-1 Program runs illustrated by example of coordination with WAITM(), unconditional wait

**References:** /PA/, Programming Guide

## SW 4

**Objective**

Decelerating and waiting must take place only in cases where not all the channels to be coordinated have set their mark numbers for the purpose of synchronization. Conditional waiting.

The instants in time for generating wait marks and the conditional wait calls are decoupled.

For the purpose of inter-channel communication, marks may even be set when waiting and decelerating are not intended at all. No WAITMC() command. In this case, the channel marks settings remain valid after execution of RESET and NC Start.

**Preconditions for conditional wait**

To utilize conditional wait with WAITMC() and reduced wait times, the following conditions must be fulfilled:

- Continuous-path mode G64 must be set
- Look Ahead function must be active
- Exact stop (G60, G09) **must not** be set.
  If exact stop is selected, waiting with WAITMC() corresponds to waiting with WAITM() from SW 3.

**Procedure**

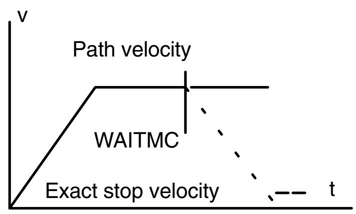
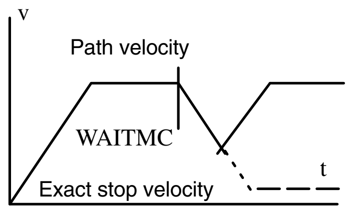
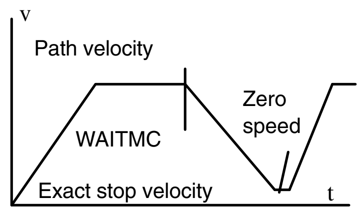
A) Starting with the motion block before the WAITMC() call, the wait marks of the other channels to be synchronized are checked. If these have all been supplied, then the channels continue to operate without deceleration in continuous-path mode. No wait. The path velocity remains unchanged.

B) If at least one wait mark from one of the channels to be synchronized is missing, then the axes start to decelerate from path velocity down to exact stop velocity. A check is now performed in every interpolation cycle to see whether the missing wait marks of the channels to be coordinated have arrived in the meantime. If this is the case, the axis is accelerated up to path velocity again and machining continued.

C) If the marks to be supplied by the channels to be synchronized have not arrived by the time exact stop velocity is reached, the machining operation is halted until the missing marks appear. When the last required mark appears, the axes are accelerated from standstill up to path velocity.

The following table shows the sequences of events for cases A) – C):

Table 2-2 Deceleration response to conditional wait with WAITMC()

| With WAITMC | Procedure | Velocity curve |
|---|---|---|
| A) Wait marks of all channels have already arrived | Continued operation with no deceleration |  |
| B) All wait marks arrived during deceleration from path velocity down to exact stop velocity | Deceleration ceases immediately when last expected wait mark appears. The axes are accelerated back up to path velocity. |  |
| C) The last wait mark does not arrive until exact stop velocity has been reached. | Brake down to exact stop velocity. When the last required mark appears, the axes are accelerated from exact stop velocity up to path velocity. |  |

### Example of conditional wait in continuous path mode

The example is schematic and shows only those commands that are relevant to the synchronization process.

**Channel 1:**

```
%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 channels 2, 3
...                              ; Processing in channel 1
N20   WAITMC(7, 2, 3)            ; Wait conditionally for mark 7 from channels
                                 ; 2 and 3
...                              ; Processing continues in channel 1
N40   WAITMC(8, 2)               ; Wait conditionally for mark 8 from channel 2
...                              ; Processing continues in channel 1
N70   M30                        ; End of channel 1
```

**Channel 2:**

```
%200
N200                             ; Processing in channel 2
N210 SETM(7)                     ; Channel 2 sets wait mark 7
...                              ; Processing continues in channel 2
N250   SETM(8)                   ; Channel 2 sets wait mark 8
N260   M30                       ; End of channel 2
```

**Channel 3:**

```
%300
N300                                  ; Processing in channel 3
...
N350   WHEN <condition> DO SETM(7)
                                      ; Set wait mark in a synchronized action
                                      ; Processing continues in channel 3
N360 M30                              ; End of channel 3
```

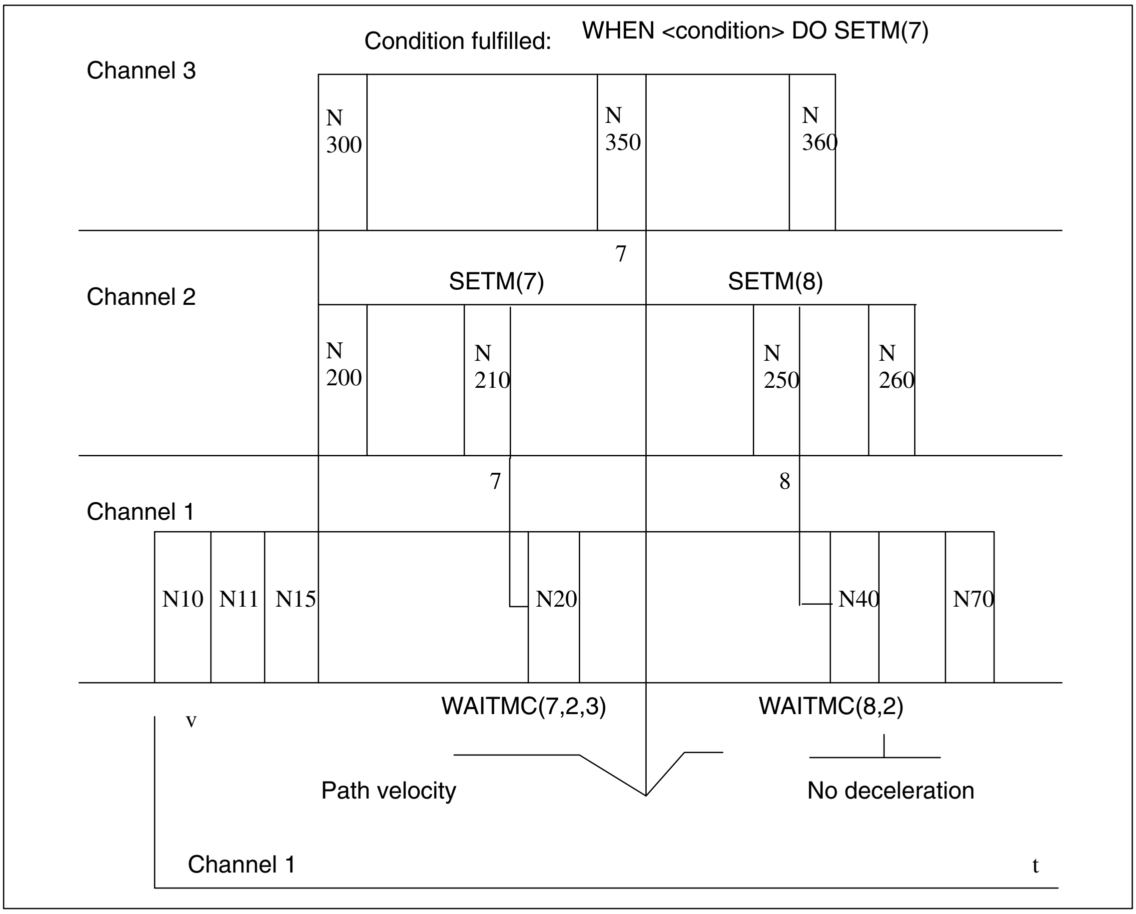



Fig. 2-2 Conditional wait involving three channels (schematic)

**WAITMC SW 5.3 and higher**

WAITMC and SETM (in Synact) can be synchronized with SW 5.3 and higher.

---

**Note**

When G64 is active, a WAITMC(1,2,3) block does not generate **a separate block**, but is appended to the preceding 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.

---

**Example of WAITMC and read-in disable**

M555 is output in channel 3 while the axis is traversing and generates a read-in disable (RID). As the WAITMC is appended to BLOCK N312, the wait mark is set and processing in channel 2 continues.

**Channel 2:**

```
N112 G18 G64 X200 Z200 F567    ; Processing in channel 2
N120 WAITMC(1,2,3)             ; Channel 2 sets wait marks 1, 2 and 3
...                            ; Further processing in channel 2 because
...                            ; WAITMC is appended to block N312.
...                            ; Further processing in channel 2
N170   M30                     ; End of channel 2
```

**Channel 3:** Read-in disable M555 during traversal

```
N300         ; Processing in channel 3
N312 G18 G64 D1 X180 Z300 M555
N320 WAITMC(1,2,3)          ; Wait because of RID
```

**Wait mark 1 is set in channels 2 and 3**

Channel 2 continues processing and program execution in channel 3 is halted due to the read-in disable.
This response can be transferred to all available channels.

**Block change in response to WAITMC SW 6.4 and higher**

With block change condition IPOBRKA, when the wait flag is received, the next block is loaded instantaneously and the axes started, provided none of the other block end conditions prevent the block change. Braking only occurs if the flag is not yet reached, or another block end condition prevents the block change.

## 2.3 Axis/spindle replacement

### 2.3.1 Introduction

**General**

An axis/spindle is firmly allocated to a certain channel via the machine data. The axis/spindle can be used in this channel only.

**Definition**

The "Axis/spindle replacement" function allows an axis or spindle to be enabled and assigned to another channel, in other words, to be replaced.

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: AUTO_GET_TYPE.

**Channel axis**

A channel axis can be programmed in the parts 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 parts 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 parts 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 parts 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**

Check and, if necessary, correct MDs 20110: RESET_MODE_MASK and MD 20112: 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 MD 30552:AUTO_GET_TYPE.
References: /FB/, K2, ...“Workpiece-Related Actual-Value System”, ...

**Displays**

The current type of axis and the current channel for this axis will be displayed in an axial PLC interface byte. See Section “Axis replacement by PLC”.

**Prerequisites**

To allow an axis to be replaced, the following must be defined via channel-specific MD 20070: AXCONF_MACHAX_USED
(machine axis number valid in channel)
**and** via
axis-specific MD 30550: AXCONF_ASSIGN_MASTER_CHAN
(initial setting of the channel for axis replacement):

1) In which channel can the axis be used and replaced?

2) To which channel shall the axis be allocated with power ON?

Example

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 MD must be allocated with:

CHANDATA(1)
AXCONF_MACHAX_USED=(**1**, 2, 3, 4, 0, 0, 0, 0)

CHANDATA(2)
AXCONF_MACHAX_USED=(5, 6, **1**, 0, 0, 0, 0, 0)

The axis-specific MD must be allocated with:

AXCONF_ASSIGN_MASTER_CHAN[AX1]=2

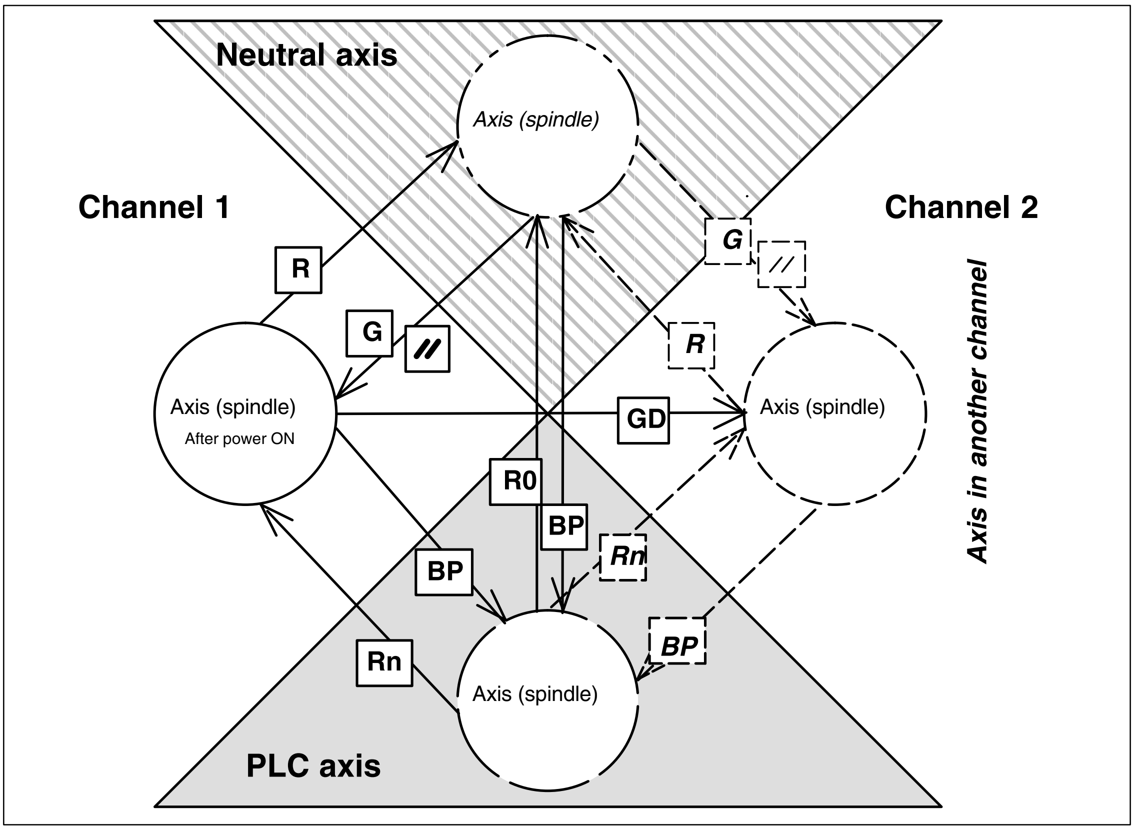
**Note**

If an axis is not valid in the channel selected, this is displayed by inversion of the axis name on the MMC.

## 2.3.2 Description

**Possible transitions**

The following diagram shows which axis replacement possibilities are available.




| | | | |
|---|---|---|---|
| R | RELEASE(AX ...) from NC program | G | GET (AX ...) from NC program |
| // | RESET key | | |
| R0 | Release in neutral state via PLC | BP | GET via PLC |
| Rn | Release to a spec. channel via PLC | GD | GET directly from NC program |

## 2.3.3 Axis transfer to neutral state (release)

**RELEASE**

Notation in parts program:
RELEASE (axis name, axis name, SPI (spindle no.), ....)

**Note**

The axis name corresponds to the axis allocations in the system and is either

- AX1, AX2, AX3, ... **or**
- the name assigned via MD 10000: AXCONF_MACHAX_NAME_TAB.

With RELEASE (axis name, ...) a dedicated NC block will always be generated.

Exception: The axis is already in the neutral state.

The RELEASE command is interrupted if

- the prerequisites for axis replacement are not fulfilled
  (MD 20070: AXCONF_MACHAX_USED and
  MD 30550: AXCONF_ASSIGN_MASTER_CHAN)
- the axis is involved in a transformation
- the axis is within an axis network

**Note**

If the RELEASE command is applied to a gantry master axis, all following axes are released, too.

| If there is ... | and ... | then ... |
|---|---|---|
| the axis is released, but not yet transferred with GET ... | ... a RESET takes place via the operator panel front ... | ... the axis is allocated again to the last responsible channel. |

## 2.3.4 Takeover of axis or spindle

**Options**

The release time and the response of an axis or spindle replacement can be controlled as follows:

- Programming in the parts program with the GET command
- Directly from another channel with GETD
- Through programming the axis name via MD 30552: AUTO_GET_TYPE, a GET or a GETD will be generated automatically.
- **With effect from SW 5.3**, it will be possible to vary the axis replacement response and the release time via MD 10722: AXCHANGE_MASK.

**Per command in the parts program GET 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 search stop is always generated.

Exception:
- 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 GETD1. If the axis is a PLC axis, replacement is delayed until the PLC has enabled the axis.

**Caution**

This programming 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 there is ... | and ... | then ... |
|---|---|---|
| the GET command has been programmed, transfer is delayed ... | ... a RESET takes place in the channel ... | ... the channel does not try any longer to take over the axis. |

An axis assumed 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 MD 30550: AXCONF_ASSIGN_MASTER_CHAN.

**Automatically through programming of axis name**

Depending on the setting in MD 30552: AUTO_GET_TYPE, a GET or GETD command is automatically generated when a neutral axis is programmed again.

**Example 1**

```
N1   M3 S1000
N2   RELEASE (SPI(1))   ;=>Transition to neutral state
N3   S3000              ; New speed for released spindle
                        ; MD AUTO_GET_TYPE    =
                        ; 0 =>Alarm "Wrong axis type" is output
                        ; 1 => GET (SPI(1)) is generated.
                        ; 2 => GETD (SPI(1)) is generated.
```

**Example 2**

```
                        ; (axis 1 = X)
N1   RELEASE (AX1)      ;=>Transition to neutral state
N2   G04 F2
N3   G0 X100 Y100:      ; Motion of released axis
                        ; MD AUTO_GET_TYPE    =
                        ; 0 =>Alarm "Wrong axis type" is output
                        ; 1 => GET (AX1) is generated.
                        ; 2 => GETD (AX1) is generated.
```

**Example 3**

```
                        ; (axis 1 = X)
N1   RELEASE (AX1)      ;=>Transition to neutral state
N2   G04 F2
N3   POS (X) = 100:     ; Positioning the released axis:
                        ; MD AUTO_GET_TYPE    =
                        ; 0 =>Alarm "Wrong axis type" is output
                        ; 1 => GET (AX1) is generated. *)
                        ; 2 => GETD (AX1) is generated. *)
```

*) If the axis is still synchronized, no dedicated block will be generated.

---

**Note**

If an automatic GETD is set, the following must be observed:

1. The channels may influence one another.
   (REORG if axis is taken away)
2. With simultaneous access of several channels to an axis it is not known which channel will have the axis at the end.

---

### 2.3.5 Axis replacement extensions SW 5.3 and higher

**Varying the axis replacement**

The machine data MD 10722: AXCHANGE_MASK provides the following options for controlling the release time of the axes or the spindle:

- Bit 0 = 1 — Automatic axis replacement also occurs between two channels if WAITP has brought the axis to a neutral state (response as before).
- Bit 1 = 1 — **With effect from SW 5.3** all the axes fetched with GET or GETD into the axis container can only be replaced again after an axis container rotation.
- Bit 2 = 1 — **With effect from SW 6.4**, when an intermediate block is inserted the main run will check whether or not reorganization is required. Reorganization is only required if the axis states of this block do **not** match the current axis states.

**As of SW 5.3**

**Activating axis container rotation**

When an axis container rotation is activated, all the axis container axes that can be assigned to the channel are assigned to the channel by means of implicit GETs or GETDs. The axes can only be released again after the axis container rotation.

---

**Note**

This response does **not** apply if an axis in main run axis state (e.g. the PLC axis) is to take part in an axis container rotation, as this axis would have to give up this state for the axis container rotation.

---

**Example**

**Axis container rotation with implicit GET or GETD**

| Action channel 1 | | Action channel 2 |
|---|---|---|
| | | SPOS = 180 positioned |
| axctswe(CT 1) | ; gets spindle in channel 1<br>; and allows axis container rotation | |
| **Exception:** | The spindle is used in both channels and is also an axis in axis container CT 1. | |

**As of SW 6.4**

### Axis replacement without a 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:

- The mode for the axis or positioned spindle
- The setpoint position

The following states of a spindle in speed mode are checked:

- Speed mode
- Spindle speed
- Direction of rotation
- Gear stage
- Master spindle at constant cutting rate.

Reorganization of the following axes is forced in any case.

**Exception**

**Axis replacement with a 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.

**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 a 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 degrees and back to 1 degree, for example and again to the neutral axis, block N040 does not trigger a preprocessing stop nor a reorganization.

### 2.3.6 Examples 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.

**Task**

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**

Assignment of channel-specific MD 20070:
AXCONF_MACHAX_USED[1]=(1, **2**, 3, 4, 0, 0, 0, 0)
AXCONF_MACHAX_USED[2]=(5, 6, **2**, 0, 0, 0, 0, 0)

Assignment of axis-specific MD 30550:
AXCONF_ASSIGN_MASTER_CHAN[AX2]=1

Program in channel 1

```
...
RELEASE (AX2)
            ; Enable axis AX2
INIT (2, "_N_MPF_DIR\_N_TAUSH2_MPF", "S")
            ; Select program TAUSCH2 in channel 2
START (2)
            ; Start program in channel 2
WAITM (1,1,2)
            ; Wait for wait mark 1 in channels 1 and 2
...         ;
...         ; Further sequence of operations after axis
            ; replacement
...         ;
...         ;
M30
```

Program TAUSH2 in channel 2

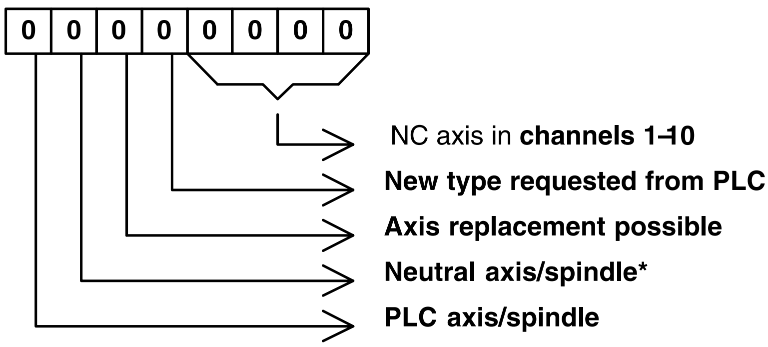
```
...
WAITM (1,1,2)
            ; Wait for wait mark 1 in channels 1 and 2
GET (AX2)
            ; Transfer axis AX2
...         ;
...         ; Further sequence of operations after axis
            ; replacement
...         ;
RELEASE (AX2)
            ; Release for further axis replacement
...         ;
M30
```

### 2.3.7 Axis replacement via PLC

- The type of an axis can be determined at any time via an interface byte (PLC axis, channel axis, neutral axis)

**TYPE display**

**NCK => PLC (DB31–..., DBB68)**




Ex. 1: Channel axis K2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Ex. 2: PLC axis. Interpolation and interface processing in K1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Ex. 3: Neutral axis. Interface processing in K3

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

* **neutral axis/spindle** also includes the **command/reciprocating axis**

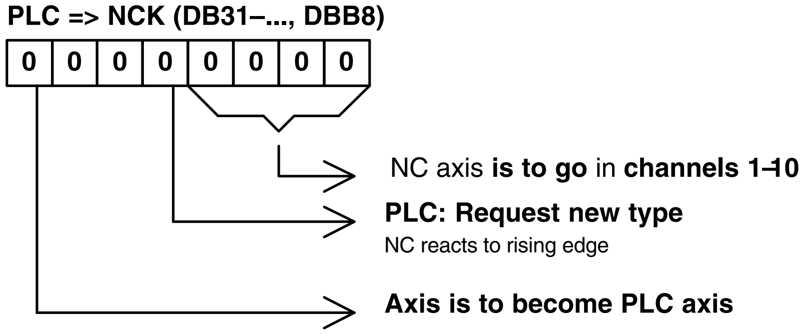
| | NCK=>PLC, DBB68 | PLC=>NCK, DBB8 |
|---|---|---|
| After power ON | 0 0 1 0 0 0 0 1 | 0 0 0 0 0 0 0 0 |
| RELEASE (K1) | 0 1 1 0 0 0 0 1 | |
| | 0 1 0 0 0 0 0 1 | |
| GET (K2) | 0 0 1 0 0 0 1 0 | |
| | 0 0 0 0 0 0 1 0 | |

Time

Fig. 2-3 Changing an axis from K1 to K2 via parts program.

- The PLC can request and traverse an axis at any time and in any operating mode.

**Specifying TYPE**




In principle, the PLC must set the signal “Request new type”. It is deleted again after change. This also applies to a channel change with GET and RELEASE.

- The PLC can change an axis from one channel to another.

PLC axes and PLC spindles are traversed via special function modules in the basic PLC program.

**FC15:** POS_AX Positioning of linear and rotary axes

**FC16:** PART_AX Positioning of indexing axes

**FC18:** SpinCtrl Spindle control

**Examples**

The following diagrams show the IS signal sequences for changing an NC axis to a PLC axis and transferring an NC axis to a neutral axis through the PLC.

| | 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

Fig. 2-4 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

Fig. 2-5 Changing an NC axis to a neutral axis through the PLC.

■

# Notes

# Supplementary Conditions 3

**“Mode group” function**

On SINUMERIK 840D up to 10 mode groups.

**Number of channels**

Up to 10 channels are available on the SINUMERIK 840D control.

**“Axis/spindle replacement” function**

This function is available for

- SINUMERIK 840D with NCU 572/573, SW2 and higher

**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 mode 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 | Destination | |

■

# Data Descriptions (MD, SD) 4

## 4.1 General machine data

| | | | |
|---|---|---|---|
| **10722**<br>MD number | **AXCHANGE_MASK**<br>Parameterization of the axis replacement response | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 0xFFFF | |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – | |
| Data type: DWORD | Applies from SW: 5.3 Bit 2 from SW 6.4 | | |
| Meaning: | This machine data can be used to change the axis replacement response.<br>Bit = 1<br>Bit 0 an automatic axis replacement via channels also occurs if WAITP has brought the axis to a neutral state.<br>Bit 1 an axctswe fetches all the axis container axes that can be assigned to the channel by means of implicit GETs or GETDs.<br>An axis can only be replaced again after the axis container rotation.<br>Bit 2 a GET generates an intermediate block without a preprocessing stop and it is only in the main run that the system checks whether reorganization is required. | | |
| Related to .... | MD: AXCONF_MACHAX_USED | | |

## 4.2 Axis/spindle-specific machine data

| | | | | | |
|---|---|---|---|---|---|
| **30550**<br>MD number | **AXCONF_ASSIGN_MASTER_CHAN**<br>Reset position of channel for axis change | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 2 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 2 | |
| Meaning: | Definition of the channel to which the axis is allocated after power ON | | | | |
| Application example(s) | With the function “Axis/spindle replacement”, a machine axis must be allocated to a channel after power on.<br>AXCONF_ASSIGN_MASTER_CHAN[AX2]=1 ⇒ axis AX2 is assigned to channel 1 after power on. | | | | |
| Related to .... | MD: AXCONF_MACHAX_USED | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **30552**<br>MD number | **AUTO_GET_TYPE**<br>Definition for automatic GET | | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: |
| Data type: BYTE | | | | Applies from SW: 3 | |
| Meaning: | 0=No automatically generated GET⇒Alarm in response to programming error.<br>1=a GET is output when GET is generated automatically.<br>2=a GETD is output when GET is generated automatically. | | | | |
| Related to .... | | | | | |

■

# 5 Signal Descriptions

| DB31, ...<br>**DBB8**<br>Data Block | **Axis/spindle replacement**<br>Signal(s) to channel (PLC ——> NCK) | |
|---|---|---|
| Edge evaluation: yes | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | The current axis type and currently active channel for this axis must be specified.<br>With axis replacement by the PLC, meaning of signal to axis/spindle DB31, ... DBB8:<br>Bit 0: A Assign NC axis/spindle to channel<br>Bit 1: B ...<br>Bit 2: C ...<br>Bit 3: D Assign NC axis/spindle to channel<br>Bit 4: Activation, assignment by positive edge<br>Bit 5: –<br>Bit 6: –<br>Bit 7: Request PLC axis/spindle | |
| Signal state 0 or signal transition 1 ——> 0 | | |
| Corresponding to... | IS DB31, ... DBB68, “Axis/spindle replacement”<br>MD 20070: AXCONF_ASSIGN_MASTER_USED (machine axis number valid in channel)<br>MD 30550: AXCONF_ASSIGN_MASTER_CHAN (initial setting of channel for axis replacement) | |
| Special cases, errors, ...... | | |

| DB31, ...<br>**DBB68**<br>Data Block | **Axis/spindle exchange**<br>Signal(s) to channel (NCK ——> PLC) | |
|---|---|---|
| Edge evaluation: yes | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | The current axis type and current channel assignment for this axis are displayed.<br>With axis exchange by the PLC, meaning of signal from axis/spindle DB31, ... DBB68:<br>Bit 0 A NC axis/spindle in channel<br>Bit 1: B ...<br>Bit 2: C ...<br>Bit 3: D NC axis/spindle in channel<br>Bit 4: New type requested by PLC<br>Bit 5: Axis can be exchanged<br>Bit 6: Neutral axis/spindle and command/oscillation axes<br>Bit 7: PLC axis/spindle | |
| Signal state 0 or signal transition 1 ——> 0 | | |
| Related to... | IS DB31, ... DBB8, “Axis/spindle replacement”<br>MD 20070: AXCONF_ASSIGN_MASTER_USED (machine axis number valid in channel)<br>MD 30550: AXCONF_ASSIGN_MASTER_CHAN (initial setting of channel for axis replacement) | |
| Special cases, errors, ...... | | |

# Notes

# 6 Example

None

# 7 Data Fields, Lists

**Reference** There is a reference for data which are not described in this Description of Functions (e.g. /K1/ means that the description can be found in Description of Functions K1).

## 7.1 General machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ...** | | | |
| 10010 | ASSIGN_CHAN_TO_MODE_GROUP[n] | Channel valid in mode group [channel no.]: 0, 1 | K1 |
| 10722 | AXCHANGE_MASK | Parameterization of the axis replacement response with effect from SW 5.3 | |

## 7.2 Channel machine data

### 7.2.1 Basic machine data of channel

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 20000 | CHAN_NAME | Channel name | K1 |
| 20050 | AXCONF_GEOAX_ASSIGN_TAB[n] | Assignment between geometry axis and channel axis<br>[GEOaxis no.]: 0...2 | K2 |

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 20060 | AXCONF_GEOAX_NAME_TAB[n] | Geometry axis name in channel [GEOaxis no.]: 0...2 | K2 |
| 20070 | AXCONF_MACHAX_USED[n] | Machine axis number valid in channel [channel axis no.]: 0...7 | K2 |
| 20080 | AXCONF_CHANAX_NAME_TAB[n] | Channel axis name in channel [channel axis no.]: 0...7 | K2 |
| 20090 | SPIND_DEF_MASTER_SPIND | Initial setting of master spindle in channel | S1 |
| 20100 | DIAMETER_AX_DEF | Geometry axis with transverse axis function | P1 |
| 20150 | GCODE_RESET_VALUES[n] | Initial setting of G groups [G group no.]: 0...59 | K1 |
| 20160 | CUBIC_SPLINE_BLOCKS | Number of blocks for C spline | K1 |
| 20170 | COMPRESS_BLOCK_PATH_LIMIT | Maximum traversing length of NC block for compression | K1 |
| 20200 | CHFRND_MAXNUM_DUMMY_BLOCKS | Empty blocks with phase/radii | K1 |
| 20210 | CUTCOM_CORNER_LIMIT | Max. angle for intersection calculation with tool radius compensation | W1 |
| 20220 | CUTCOM_MAX_DISC | Maximum value with DISC | W1 |
| 20230 | CUTCOM_CURVE_INSERT_LIMIT | Maximum angle for intersection calculation with tool radius compensation | W1 |
| 20240 | CUTCOM_MAXNUM_CHECK_BLOCKS | Blocks for predictive contour calculation with tool radius compensation | W1 |
| 20250 | CUTCOM_MAXNUM_DUMMY_BLOCKS | Max. no. of dummy blocks with no traversing movements | W1 |
| 20270 | CUTTING_EDGE_DEFAULT | Basic setting of tool cutting edge without pro-gramming | W1 |
| 20400 | LOOKAH_USE_VELO_NEXT_BLOCK | Look Ahead to programmed following block velocity | B1 |
| 20430 | LOOKAH_NUM_OVR_POINTS | Number of override switch points for Look Ahead | B1 |
| 20440 | LOOKAH_OVR_POINTS[n] | Prepared override velocity characteristics with Look Ahead [characteristic no.]: 0...1 | B1 |
| 20500 | CONST_VELO_MIN_TIME | Minimum time with constant velocity | B2 |
| 20600 | MAX_PATH_JERK | Pathrelated maximum jerk | B2 |
| 20610 | ADD_MOVE_ACCEL_RESERVE | Acceleration reserve for overlaid movements | K1 |
| 20650 | THREAD_START_IS_HARD | Acceleration behavior of axis with thread cut-ting | K1 |
| 20700 | REFP_NC_START_LOCK | NC start disable without reference point | R1 |
| 20750 | ALLOW_GO_IN_G96 | G0 logic in G96 | V1 |
| 20800 | SPF_END_TO_VDI | Subprogram end to PLC | H2 |
| 21000 | CIRCLE_ERROR_CONST | Circle end point monitoring constant | K1 |
| 21010 | CIRCLE_ERROR_FACTOR | Circle end point monitoring factor | K1 |
| 21100 | ORIENTATION_IS_EULER | Angle definition for orientation programming | F2 |
| 21110 | X_AXIS_IN_OLD_X_Z_PLANE | Coordinate system for automatic Frame defini-tion | K2 |

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 21200 | LIFTFAST_DIST | Traversing path for fast retraction from the contour | K1 |
| 21250 | START_INDEX_R_PARAM | Number of first channelspecific R parameter | S7 |

## 7.2.2 Auxiliary function settings of channel

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 22000 | AUXFU_ASSIGN_GROUP[n] | Auxiliary function group [aux. func. no. in channel]: 0...49 | H2 |
| 22010 | AUXFU_ASSIGN_TYPE[n] | Auxiliary function type [aux. func. no. in channel]: 0...49 | H2 |
| 22020 | AUXFU_ASSIGN_EXTENSION[n] | Auxiliary function extension [aux. func. no. in channel ]: 0...49 | H2 |
| 22030 | AUXFU_ASSIGN_VALUE[n] | Auxiliary function value [aux. func. no. in channel ]: 0...49 | H2 |
| 22200 | AUXFU_M_SYNC_TYPE | Output timing for M functions | H2 |
| 22210 | AUXFU_S_SYNC_TYPE | Output timing of S functions | H2 |
| 22220 | AUXFU_T_SYNC_TYPE | Output timing of T functions | H2 |
| 22230 | AUXFU_H_SYNC_TYPE | Output timing for H functions | H2 |
| 22240 | AUXFU_F_SYNC_TYPE | Output timing of F functions | H2 |
| 22250 | AUXFU_D_SYNC_TYPE | Output timing of D functions | H2 |
| 22260 | AUXFU_E_SYNC_TYPE (available soon) | Output time of E functions. | – |
| 22300 | AUXFU_AT_BLOCK_SEARCH_END | Output of auxiliary functions after block search | H2 |
| 22400 | S_VALUES_ACTIVE_AFTER_RESET | S function active after RESET | S1 |
| 22410 | F_VALUES_ACTIVE_AFTER_RESET | F function active after reset | V1 |
| 22500 | GCODE_OUTPUT_TO_PLC | G functions to PLC | K1 |
| 22550 | TOOL_CHANGE_MODE | New tool offset for M function | W1 |
| 22560 | TOOL_CHANGE_M_CODE | M function for tool change | W1 |

## 7.2.3 Transformation definitions in channel

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 24100 | TRAFO_TYPE_1 | Definition of transformation 1 in channel | F2 |
| 24110 | TRAFO_AXES_IN_1[n] | Axis assignment for transformation [axis index]: 0...7 | F2 |

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 24120 | TRAFO_GEOAX_ASSIGN_TAB_1[n] | Assignment between GEO axis and channel axis for transformation 1 [GEO axis no.]: 0...2 | F2 |
| 24200 | TRAFO_TYPE_2 | Definition of transformation 2 in channel | F2 |
| 24210 | TRAFO_AXES_IN_2[n] | Axis assignment for transformation 2 [axis index]: 0...7 | F2 |
| 24220 | TRAFO_GEOAX_ASSIGN_TAB_2[n] | Assignment between GEO axis and channel axis for transformation 2 [GEO axis no.]: 0...2 | F2 |
| 24300 | TRAFO_TYPE_3 | Definition of transformation 3 in channel | F2 |
| 24310 | TRAFO_AXES_IN_3[n] | Axis assignment for transformation 3 [axis index]: 0...7 | F2 |
| 24320 | TRAFO_GEOAX_ASSIGN_TAB_3[n] | Assignment between GEO axis and channel axis for transformation 3 [GEO axis no.]: 0...2 | F2 |
| 24400 | TRAFO_TYPE_4 | Definition of transformation 4 in channel | F2 |
| 24410 | TRAFO_AXES_IN_4[n] | Axis assignment for transformation 4 [axis index]: 0...7 | F2 |
| 24420 | TRAFO_GEOAX_ASSIGN_TAB_4[n] | Assignment between GEO axis and channel axis for transformation 4 [GEO axis no.]: 0...2 | F2 |
| 24430 | TRAFO_TYPE_5 | Definition of transformation 5 in channel | F2 |
| 24432 | TRAFO_AXES_IN_5[n] | Axis assignment for transformation 5 [axis index]: 0...7 | F2 |
| 24434 | TRAFO_GEOAX_ASSIGN_TAB_5[n] | Assignment between GEO axis and channel axis for transformation 5 [GEO axis no.]: 0...2 | F2, M1 |
| 24440 | TRAFO_TYPE_6 | Definition of transformation 6 in channel | F2 |
| 24442 | TRAFO_AXES_IN_6[n] | Axis assignment for transformation 6 [axis index]: 0...7 | F2 |
| 24444 | TRAFO_GEOAX_ASSIGN_TAB_6[n] | Assignment between GEO axis and channel axis for transformation 6 [GEO axis no.]: 0...2 | F2, M1 |
| 24450 | TRAFO_TYPE_7 | Definition of transformation 7 in channel | F2 |
| 24452 | TRAFO_AXES_IN_7[n] | Axis assignment for transformation 7 [axis index]: 0...7 | F2 |
| 24454 | TRAFO_GEOAX_ASSIGN_TAB_7[n] | Assignment between GEO axis and channel axis for transformation 7 [GEO axis no.]: 0...2 | F2, M1 |
| 24460 | TRAFO_TYPE_8 | Definition of transformation 8 in channel | F2 |
| 24462 | TRAFO_AXES_IN_8[n] | Axis assignment for transformation 8 [axis index]: 0...7 | F2 |
| 24464 | TRAFO_GEOAX_ASSIGN_TAB_8[n] | Assignment between GEO axis and channel axis for transformation 8 [GEO axis no.]: 0...2 | F2, M1 |
| 24500 | TRAFO5_PART_OFFSET_1[n] | Offset vector of 5axis transformation 1 [axis no.]: 0...2 | F2 |
| 24510 | TRAFO5_ROT_AX_OFFSET_1[n] | Position offset of rotary axes 1/2 for 5axis transformation 1 [axis no.]: 0...1 | F2 |
| 24520 | TRAFO5_ROT_SIGN_IS_PLUS_1[n] | Sign of rotary axis 1/2 for 5-axis transformation 1 [axis no.]: 0...1 | F2 |
| 24530 | TRAFO5_NON_POLE_LIMIT_1 | Definition of pole limit for 5-axis transformation 1 | F2 |
| 24540 | TRAFO5_POLE_LIMIT_1 | Pole end angle tolerance for interpolation for 5axis interpolation 1 | F2 |

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 24550 | TRAFO5_BASE_TOOL_1[n] | Vector of base tool with activation of 5-axis transformation 1 [axis no.]: 0...2 | F2 |
| 24560 | TRAFO5_JOINT_OFFSET_1[n] | Vector of kinematic offset for 5-axis transformation 1 [axis no.]: 0...2 | F2 |
| 24600 | TRAFO5_PART_OFFSET_2[n] | Offset vector of 5axis transformation 2 [axis no.]: 0...2 | F2 |
| 24610 | TRAFO5_ROT_AX_OFFSET_2[n] | Position offset of rotary axes 1/2 for 5axis transformation 2 [axis no.]: 0...1 | F2 |
| 24620 | TRAFO5_ROT_SIGN_IS_PLUS_2[n] | Sign of rotary axis 1/2 for 5-axis transformation 2 [axis no.]: 0...1 | F2 |
| 24630 | TRAFO5_NON_POLE_LIMIT_2 | Definition of pole limit for 5-axis transformation 2 | F2 |
| 24640 | TRAFO5_POLE_LIMIT_2 | Pole end angle tolerance for interpolation for 5axis interpolation 2 | F2 |
| 24650 | TRAFO5_BASE_TOOL_2[n] | Vector of base tool with activation of 5-axis transformation 2 [axis no.]: 0...2 | F2 |
| 24660 | TRAFO5_JOINT_OFFSET_2[n] | Vector of kinematic offset for 5-axis transformation 2 [axis no.]: 0...2 | F2 |

### 7.2.4 Channel-specific memory settings

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 25000 | REORG_LOG_LIMIT | Percentage of IPO buffer for log file enable | S7 |
| 28000 | MM_REORG_LOG_FILE_MEM | Memory size for REORG (DRAM) | S7 |
| 28010 | MM_NUM_REORG_LUD_MODULES | Number of blocks for local user variables for REORG (DRAM) | S7 |
| 28020 | MM_NUM_LUD_NAMES_TOTAL | Number of local user variables (DRAM) | S7 |
| 28030 | MM_NUM_LUD_NAMES_PER_PROG | Number of local user variables per program (DRAM) | S7 |
| 28040 | MM_LUD_VALUES_MEM | Memory size for local user variables (DRAM) | S7 |
| 28050 | MM_NUM_R_PARAM | Number of channelspecific R parameters (SRAM) | S7 |
| 28060 | MM_IPO_BUFFER_SIZE | Number of NC blocks in IPO buffer (DRAM) | S7 |
| 28070 | MM_NUM_BLOCKS_IN_PREP | Number of blocks for block preparation (DRAM) | S7 |
| 28080 | MM_NUM_USER_FRAMES | Number of settable Frames (SRAM) | S7 |

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 28090 | MM_NUM_CC_BLOCK_ELEMENTS | Number of block elements for compile cycles (DRAM) | S7 |
| 28100 | MM_NUM_CC_BLOCK_USER_MEM | Size of block memory for compile cycles (DRAM) | S7 |
| 28500 | MM_PREP_TASK_STACK_SIZE | Stack size of preparation task (DRAM) | S7 |
| 28510 | MM_IPO_TASK_STACK_SIZE | Stack size of IPO task (DRAM) | S7 |

## 7.3 Axis/spindle-specific machine data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Axis/channelspecific($MA_ ... )** | | | |
| 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 | K1 |
| 33100 | COMPRESS_POS_TOL | Maximum deviation with compensation | K1 |

## 7.4 Channel-specific setting data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Channelspecific($SC_ ... )** | | | |
| 42000 | THREAD_START_ANGLE | Start angle for thread | K1 |
| 42100 | DRY_RUN_FEED | Dry run feedrate | V1 |

# 7.5 Interface signals

## 7.5.1 Mode group signals

**Description of interface signals**

The mode group signals from PLC → NCK and from NCK → PLC are stored in data block 11 for the first mode group. The signals are displayed and described in

**References:** /FB/, K1, “Mode Group, Channels, Program Operation Mode”

## 7.5.2 Channel signals

**Description of interface signals**

The channel signals from PLC → NCK and from NCK → PLC are stored in data blocks 21, 22, ... for the first, second ... channel. The signals are displayed and described in

**References:** /FB/, K1, “Mode Group, Channels, Program Operation Mode”

# 7.6 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, Diagnostics Guide
and the online help of MMC 101/102/103 systems.

■

# Notes

# SINUMERIK 840D/840Di/810D Description of Functions Extended Functions (FB2)

# Kinematic Transformations (M1)

■

# Notes

# 1 Brief Description

## 1.1 TRANSMIT

The TRANSMIT function enables the following:

- 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 perpendicular to axis of rotation (2)
  - Longitudinal axis in parallel to 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 the rotations.
- A path in the cartesian coordinate system must not pass through the turning center point (this restriction applies to SW 2 and 3).

**Additional advantages from SW 4**

- 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.

## 1.2 TRACYL

The functional scope of TRACYL (cylinder generated surface curve transformation) is as follows:

Machine

- Longitudinal grooves on cylindrical objects,
- Transverse grooves on cylindrical objects,
- Arbitrary groove patterns on cylindrical objects.

The grooving path is programmed in relation to the developed, plane cylinder generated surface.

For machining purposes, the function supports lathes with

- X-C-Z kinematics and
- X-Y-Z-C kinematics
- 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 perpendicular to axis of rotation (2)
  - Longitudinal axis in parallel to 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 perpendicular to axis of rotation (2)
  - Longitudinal axis in parallel to 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 the rotations.

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 machines can provide redundancy in the form of an additional linear axis positioned perpendicular to the rotary axis and first linear axis, this can be utilized to improve the tool offset.

## 1.3 TRAANG

The “Inclined axis” function is provided for grinding applications.
Its functional scope is as follows:

- 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.

## 1.4 Chained transformations

**Introduction**

In SW 5 and higher two transformations can be chained such that the motion parts for the axes from the first transformation are input data for the chained second transformation. The motion parts from the second transformation act on the machine axes.

**Chaining options**

- In SW 5 the chain may encompass **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 details about chained transformations, please refer to Section 2.4, and for further information about other transformations to

**References:** /FB/, F2, "3 to 5-Axis Transformations"

## 1.5 Activating the transformation MD via parts program/softkey (SW 5.2 and later)

**As of SW 5.2**

Most of the machine data relevant to kinematic transformations were activated by power ON in earlier versions.

In SW 5.2 and higher, you can also activate transformations MDs via the part program/softkey and it is not necessary to boot the control.

Please refer to Section 2.7 for a detailed description.

■

# Detailed Description 2

## 2.1 TRANSMIT

---

**Note**

The TRANSMIT transformation described below requires that individual names are assigned to machine axes, channels and geometry axes when the transformation is active.
Cf.
MD 10000: AXCONF_MACHAX_NAME_TAB,
MD 20080: AXCONF_CHANAX_NAME_TAB,
MD 20060: AXCONF_GEOAX_NAME_TAB.
This is the only method of ensuring unique assignments.

---

**Task assignment**

Complete machining, see diagram.

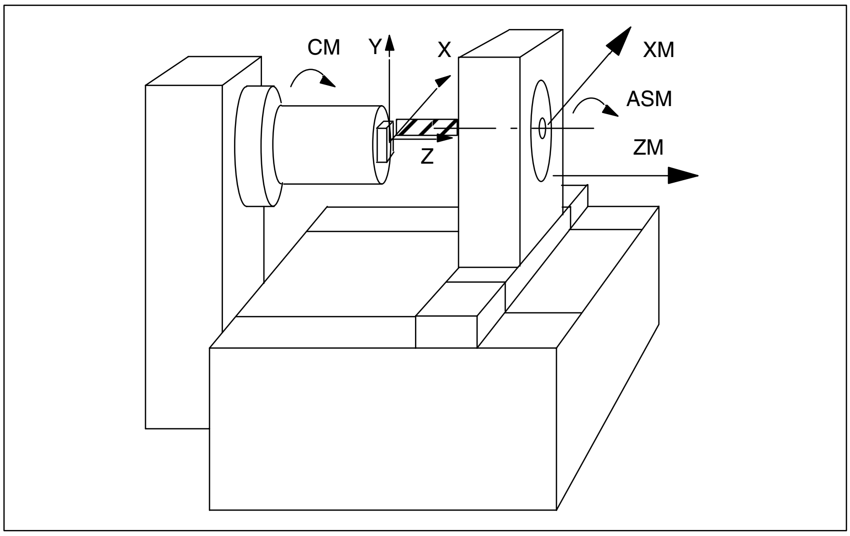



Fig. 2-1 End face machining of turned part

Legend:

| | |
|---|---|
| CM | Rotary axis (main spindle) |
| ASM | Working spindle (milling cutter, drill) |
| X, Y, Z | Cartesian coordinate system for programming the face-end machining operation (origin in turning center point of face end) |
| ZM | Machine axis (linear) |
| XM | Machine axis (linear) |

### 2.1.1 Preconditions for TRANSMIT

**Axis configuration**

Before movements can be programmed in the Cartesian coordinate system (acc. to Fig. 2-1 X, Y, Z), the control system must be notified of the relationship between this coordinate system and the real machine axes (CM, XM, ZM, ASM):

- Assignment of names to geometry axes
- Assignment of geometry axes to channel axes
  - General situation (TRANSMIT not active)
  - TRANSMIT active
- Assignment of channel axes to machine axis numbers
- Identification of spindles
- Allocation of machine axis names

With the exception of the “– TRANSMIT active” point, the procedure is the same as for the normal axis configuration. If you already know the general steps, you need only read step “Assignment of geometry axes to channel axes” from the list of steps below.

**References:** /FB/, K2, “Coordinate Systems, Axis Types, Axis Configurations, Actual-Value System for Workpiece, External Zero Offset”

**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. The following sections include descriptions of these data:

$MC_TRAFO_TYPE_n
$MC_TRAFO_GEOAX_ASSIGN_TAB_n
$MC_TRAFO_AXES_IN_n

**Number of TRANSMIT structures**

Two of the 10 permitted data structures for transformations in the channel may be assigned to the TRANSMIT function. They are characterized by the fact that the value assigned with $MC_TRAFO_TYPE_n is 256 or 257.
For these 2 TRANSMIT transformations, the following machine data must be set in a defined way:

$MC_TRANSMIT_ROT_AX_OFFSET_t
$MC_TRANSMIT_ROT_SIGN_IS_PLUS_t
$MC_TRANSMIT_BASE_TOOL_t
$MC_TRANSMIT_POLE_SIDE_FIX_t

In this case, t specifies the number of the declared TRANSMIT transformation (maximum of 2).

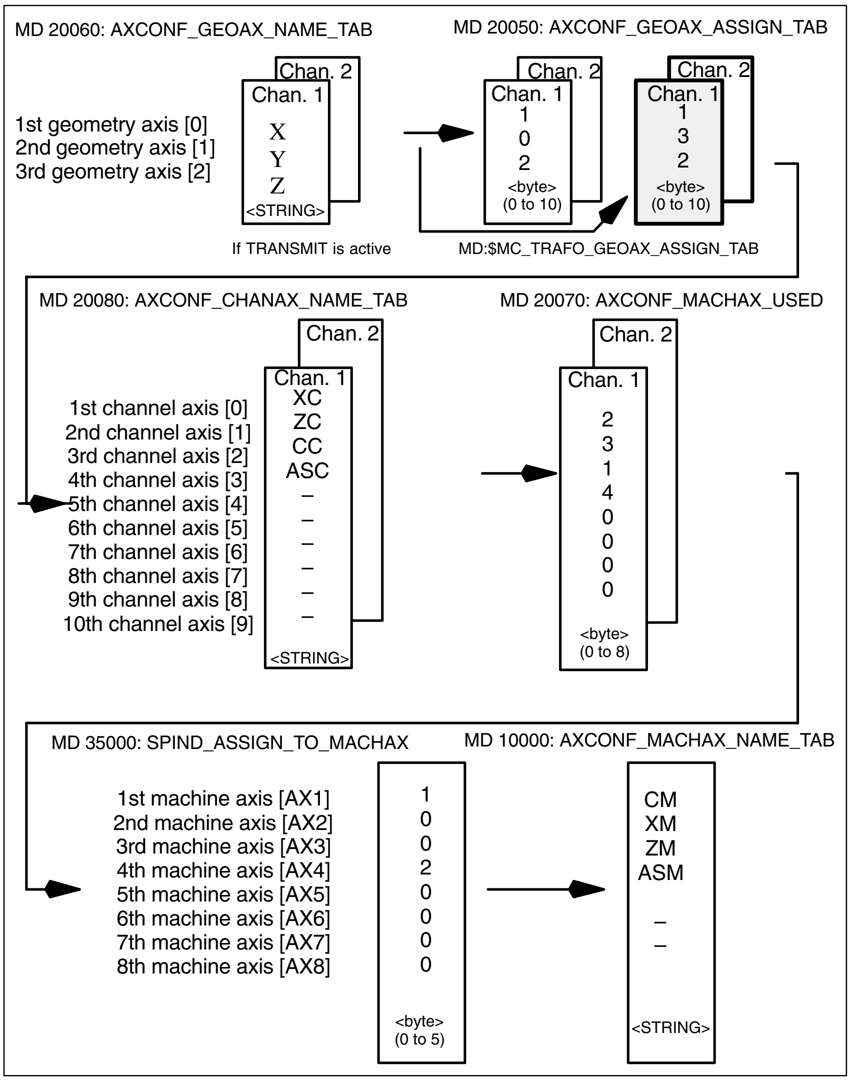



Fig. 2-2 Axis configuration for the example in Fig. 2-1

The configurations highlighted in Fig. 2-2 apply when TRANSMIT is active.

**Assignment of names to geometry axes**

According to the axis configuration overview shown above, the geometry axes to be involved in the TRANSMIT operation must be defined with:

$MC_AXCONF_GEOAX_NAME_TAB[0]="X"
" _TAB[1]="Y"
" _TAB[2]="Z"
(name selection according to Fig. 2-2, also corresponds to default setting)

**Assignment of geometry axes to channel axes**

These assignments are made depending on whether or not TRANSMIT is active.

– TRANSMIT not active
  A Y axis is not available.

```
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]=1
                    ”           _TAB[1]=0
                    ”           _TAB[2]=2
```

– TRANSMIT active
  The Y axis can be addressed with the part program.

```
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
                    ”          _TAB_1[1]=3
                    ”          _TAB_1[2]=2
```

The Y axis is the third entry for the channel axes.

**Entry of channel axes**

Axes which do not belong to the Cartesian coordinate system are entered.

```
$MC_AXCONF_CHANAX_NAME_TAB[0]=“XC”
                ”             [1]=“ZC”
                ”             [2]=“CC”
                ”             [3]=“ASC”
```

**Assignment of channel axes to machine axes**

With the cd of the channel axes as a reference, the machine axis number to which the channel axes have been assigned is transferred to the control system.

```
$MC_AXCONF_MACHAX_USED[0]=2
                ”         [1]=3
                ”         [2]=1
                ”         [3]=4
                ”         [4]=0
```

(entries corresponding to Fig. 2-2)

**Identification of spindles**

The user defines whether each machine axis is a spindle (value > 0: spindle number) or a path axis (value 0).

```
$MA_SPIND_ASSIGN_TO_MACHAX[0]=1
                ”            [1]=0
                ”            [2]=0
                ”            [3]=2
```

**Assignment of names to machine axes**

With the cd of the machine axes as a reference, a machine axis name is transferred to the control system.

```
$MN_AXCONF_MACHAX_NAME_TAB[0]=“CM”
                ”             [1]=“XM”
                ”             [2]=“ZM”
                ”             [3]=“ASM”
```

### 2.1.2 Settings specific to TRANSMIT

**Type of transformation**

The following paragraph describes how the transformation type is specified.

**TRAFO_TYPE_n**

The user must specify the transformation type for the transformation data blocks (maximum n = 10). The value 256 must be set for TRANSMIT or the VALUE 257 for a rotary axis with supplementary linear axis.

Example of VALUE 256: MD 24100: TRAFO_TYPE_1=256

The setting must be made before TRANSMIT or TRANSMIT(t) is called. “t” is the number of the declared TRANSMIT transformation.

The TRANSMIT transformation requires only a rotary axis and a linear axis positioned perpendicular to the rotary axis. A real Y axis is used with transformation type 257 in order, for example, to compensate a tool offset.

**Transformation type 257**

**Polar transformation with a rotary axis TRAFO_TYPE_n = 257**

**With supplementary linear axis**

If the machine has another linear axis which is perpendicular to both the rotary axis and the first linear axis, transformation type 257 can be used to apply tool offsets with the real Y axis. In this case, it is assumed that the user memory of the second linear axis is small and will not be used to execute the part program.

The existing settings for $MC_TRAFO_GEOAX_ASSIGN_TAB_n apply.

**Axis image**

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

**TRAFO_AXES_IN_n**

Three channel axis numbers must be specified for the transformation data block n:

| | |
|---|---|
| $MC_TRAFO_AXES_IN_1[0]= | Channel axis number of axis perpendicular to rotary axis. |
| $MC_TRAFO_AXES_IN_1[1]= | Channel axis number of rotary axis |
| $MC_TRAFO_AXES_IN_1[2]= | Channel axis number of axis parallel to rotary axis |

Example of the configuration in Fig. 2-1:
$MC_TRAFO_AXES_IN_1[0]=1
$MC_TRAFO_AXES_IN_1[1]=3
$MC_TRAFO_AXES_IN_1[2]=2

The setting must be made before TRANSMIT or TRANSMIT(t) is called. The axis numbers must refer to the channel axis sequences defined with $MC_TRAFO_GEOAX_ASSIGN_TAB_n.

**For transformation type 257**, the following index assignments for $MC_TRAFO_AXES_IN_n[ ] must be selected.
Meaning of indices in relation to basic coordinate system (BCS):

- [0]: Cartesian axis perpendicular to rotary axis (in the machine zero position, this axis is parallel to the linear axis which is positioned perpendicular to the rotary axis)
- [1]: Cartesian axis perpendicular to rotary axis
- [2]: Cartesian axis parallel to rotary axis (if configured)
- [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 perpendicular to rotary axis
- [1]: Rotary axis
- [2]: Linear axis parallel to rotary axis (if configured)
- [3]: Linear axis perpendicular to the axes of indices [0] and [1]

**Rotational position**

The rotational position of the Cartesian coordinate system is specified by machine data as described in the following paragraph.

**TRANSMIT _ROT_AX _OFFSET_t**

The rotational position of the x-y plane of the Cartesian coordinate system in relation to the defined zero position of the rotary axis is specified with:

$MC_TRANSMIT_ROT_AX_OFFSET_t= ... ; degrees

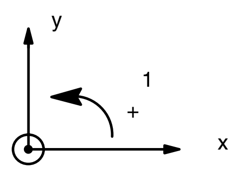
In this case, "t" is substituted by the number of the TRANSMIT transformations declared in the transformation data blocks.
(t must not be more than 2).

**Direction of rotation**

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

**TRANSMIT _ROT_SIGN _IS_PLUS_t**

If the rotary axis rotates in an anti-clockwise direction on the X-Y plane when viewed along the Z axis, then the machine axis must be set to 1, but otherwise to 0.




$MC_TRANSMIT_ROT_SIGN_IS_PLUS_t=1
In this case, "t" is substituted by the number of the TRANSMIT transformations declared in the transformation data blocks (t must not be more than 2).

**Position of tool zero point**

The position of the tool zero point is specified by machine data as described in the following paragraph.

**TRANSMIT_ BASE_TOOL_t**

Machine data $MC_TRANSMIT_BASE_TOOL_t is used to inform the control of the position of the tool zero point in relation to the origin of the coordinate system declared for TRANSMIT. The machine data has three components for the three axes of the Cartesian coordinate system.

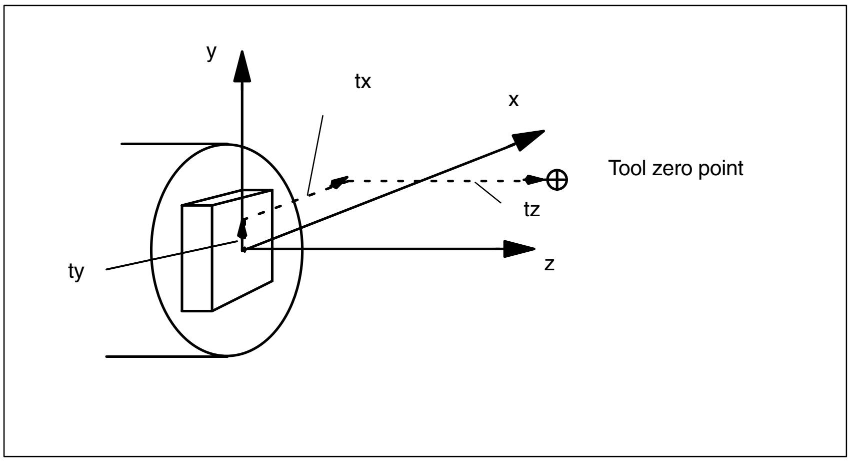



Fig. 2-3 Position of tool zero in relation to origin of the Cartesian coordinate system (see Fig. 2-1)

$MC_TRANSMIT_BASE_TOOL_t[0]=tx
” [1]=ty
” [2]=tz

In this case, t in front of the index value [ ] is replaced by the number of the TRANSMIT transformations declared in the transformation data blocks.
(t must not be more 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.

- MD 22534: 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:** /FB/, K2, “Coordinate Systems, Axis Types, Axis Configurations, Actual-Value System for Workpiece, External Zero Offset”

### 2.1.3 Activation of TRANSMIT

**TRANSMIT**

After the settings described above have been made, the TRANSMIT function can be activated:

TRANSMIT or
TRANSMIT(t)

The first declared TRANSMIT function is activated with TRANSMIT.
TRANSMIT(t) activates the t-th declared TRANSMIT function. t may not be more than 2.
When TRANSMIT is activated in SW 4 and higher, the special processes for pole traversal, etc. according to 2.1.6 become available.

Between activation of the function and deactivation as described below, the traversing movements for the axes of the Cartesian coordinate system can be programmed.

### 2.1.4 Deactivation of TRANSMIT function

**TRAFOOF**

Keyword TRAFOOF deactivates an active transformation. When the transformation is deactivated, the base coordinate system is again identical to the machine coordinate system.
An active TRANSMIT transformation is likewise deactivated if one of the other transformations is activated in the relevant channel.
(e.g. TRACYL, TRAANG, TRAORI).

**References:** /FB/, F2, “3-5-Axis Transformation”

### 2.1.5 Special system reactions with TRANSMIT

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

**Please note on selection**

- An intermediate motion block is not inserted (phases/radii).
- A spline block sequence must be terminated.
- Tool radius compensation must be deselected.
- An activated tool length compensation is incorporated into the transformation in the geometry axis by the control.
- The frame which was active prior to TRANSMIT is deselected by the control. (Corresponds to Reset programmed frame G500).
- An active working area limitation is deselected by the control for the axes affected by the transformation (corresponds to programmed WALIMOF).
- Continuous path control and rounding are interrupted.
- DRF offsets in transformed axes must have been deleted by the operator.

**Please note on deselection**

- An intermediate motion block is not inserted (phases/radii).
- A spline block sequence must be terminated.
- Tool radius compensation must be deselected.
- The frame which was active prior to TRANSMIT is deselected by the control (corresponds to Reset programmed frame G500).
- Continuous path control and rounding are interrupted.
- DRF offsets in transformed axes must have been deleted by the operator.
- Tool length compensation in the virtual axis (the Y axis in Fig. 2-1) is not implemented.

**Restrictions when TRANSMITis active**

The restrictions listed below imposed by an activated TRANSMIT function must be noted.

**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.

**Rotary axis**

The rotary axis cannot be programmed because it is occupied by a geometry axis and cannot thus be programmed directly as a channel axis.

**Extensions with SW 6.4 and later**
An offset in the rotary axis CM can be entered, for example, by compensating the inclined position of a workpiece in a frame within the frame chain. The x and y values are then as illustrated in the following diagram.

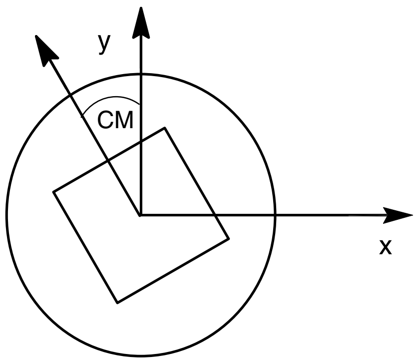



Fig. 2-4 Rotary axis offset with TRANSMIT

With SW 6.4 and later, this offset can also be included in the transformation as an offset in the rotary axis. To ensure that the total axial frame of the transmit rotary axis, i.e. the translation, fine offset, mirroring and scaling, is included in the transformation, the following settings must be made:

MD 24905: TRANSMIT_ROT_AX_FRAME_1 = 1
MD 24955: TRANSMIT_ROT_AX_FRAME_2 = 1

---

**Note**

Changes in the axis assignments are converted every time the transformation is selected or deselected. For further information about axial offsets for rotary axes to the SZS, please see:

**References:** /FB/, K2, “Coordinate Systems, Frames”

---

**Pole**

SW up to and including 3.x:
Movements through the pole (origin of Cartesian coordinate system) are disabled, i.e. a movement which traverses the pole is stopped in the pole followed by the output of an alarm. In the case of a cutter center offset, the movement is terminated accordingly at the end of the non-approachable area.
SW 4 and higher:
The options for pole traversal and machining operations close to the pole are described in the Subsections starting at 2.1.6.

**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 TRANSMIT 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 which is included in the transformation via machine data $MC_TRAFO_AXES_IN_n index 0 or 1.
- On changeover to JOG mode.

The monitoring function is transferred from the main run back to the preprocessing routine if the axes relevant to the transformation process are operated as path axes.

The velocity monitoring function in preprocessing utilizes the machine better than the monitoring in the main run. Furthermore, the main run monitoring function deactivates the Look Ahead.

**Interruption of parts program**

If parts program processing is interrupted for JOG, then the following must be noted:

**JOG**

When JOG is selected, the conventional on-line velocity check is activated instead of the optimized velocity check provided in 2.1.6 SW 4.

**From AUTOMATIC to JOG**

If parts program processing is interrupted when the transformation is active followed by traversal in JOG mode, then the following must be noted when AUTOMATIC is selected again:

- The transformation is active in the approach block from the current position to the point of interruption. No monitoring for collisions takes place.

**Warning**

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

**In AUTOMATIC mode**

The velocity-optimized velocity planning function (SW 4) remains active for as long as the axes relevant to the transformation are traversed in mutual synchronism as path axes. If an axis involved in the transformation is traversed as a positioning axis, the online velocity check remains active until the transformation is deactivated or until all axes involved in the transformation are operating as path axes again. The return to velocity-optimized operation according to 2.1.6 automatically initiates a STOPRE and synchronizes acyclic block preprocessing with the interpolation routine.

**From start after reset**

If parts program processing is aborted with RESET and restarted with START, then the following must be noted:

- The remaining parts program is traversed reproducibly only if all axes are traversed to a defined position by means of a linear block (G0 or G1) at the beginning of the parts program. A tool which was active on RESET may no longer be taken into account by the control (settable via machine data).

**Power On RESET**

The system response after power ON is determined by the settings stored in
MD 20110: RESET_MODE_MASK and
MD 20140: TRAFO_RESET_VALUE

**References:** /FB/, K2, "Workpiece-Related Actual-Value System"

**Reference point approach**

Axes cannot be referenced when a transformation is active. Any active transformation is deselected by the control system during a referencing operation.

### 2.1.6 Machining with TRANSMIT using SW 4.x and higher

**Introduction**

The TRANSMIT transformation has a pole at the zero point of the TRANSMIT plane (example, see Fig.: 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.

With SW 4 and higher, 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 examined:

- 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**

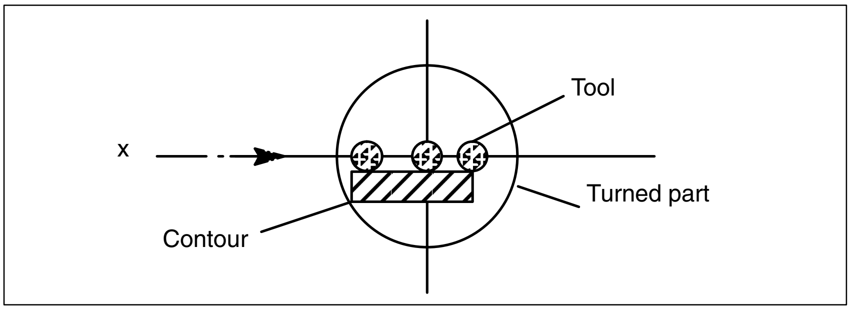



Fig. 2-5 Traversal of x axis through pole

### Rotation in pole

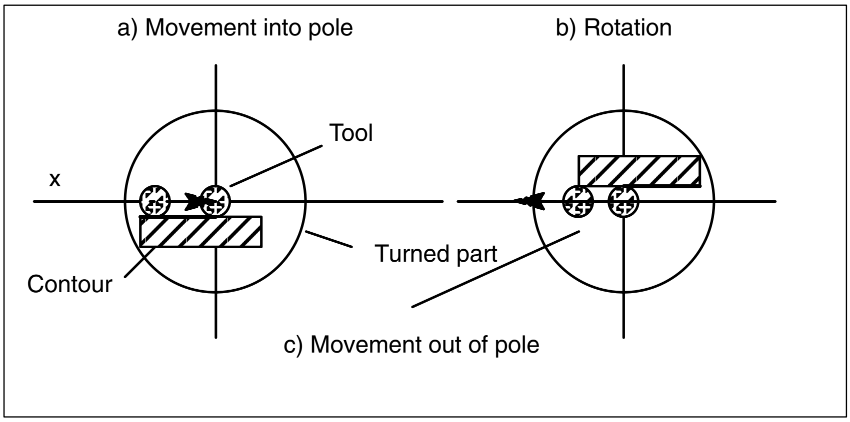



Fig. 2-6 Traversal of x axis into pole (a), rotation (b), exit from pole (c)

### 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:

MD 24911: TRANSMIT_POLE_SIDE_FIX_1
MD 24951: 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.

Table 2-1

| VALUE | Meaning |
|---|---|
| 0 | Pole traversal<br>The tool center point path (linear axis) must traverse the pole on a continuous path. |
| 1 | Rotation around pole<br>The tool center point path must be restricted to the positive traversing range of the linear axis (in front of turning center). |
| 2 | Rotation around pole<br>The tool center point path must be restricted to the 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 2-2 Traversal of pole along the linear axis

| Operating mode | Status | 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. positioning axis). TRANSMIT not active. | Traversal of pole at creep speed |
| | An applied DRF (external zero offset) does not interfere with the operation. Servo errors may occur close to the pole during application of a DRF. | Abortion of machining operation, alarm |
| JOG | – | Traversal of pole at creep speed |

### Special features relating to rotation in pole

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

MD 24911: TRANSMIT_POLE_SIDE_FIX_1 = 1 or 2
MD 24951: 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: MAX_AX_VELO[AX*] and MD 32300: 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.

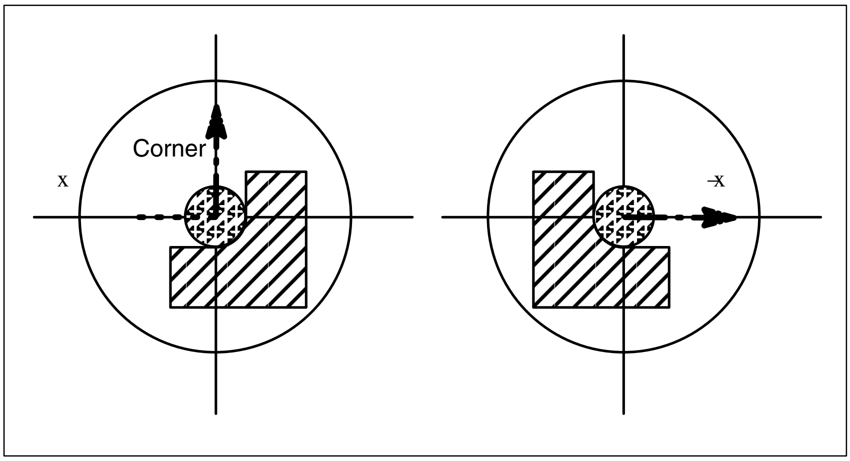



Fig. 2-7 Pole traversal

Requirements:
AUTOMATIC mode, MD 24911: TRANSMIT_POLE_SIDE_FIX_1= 0 or
MD 24951: 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**

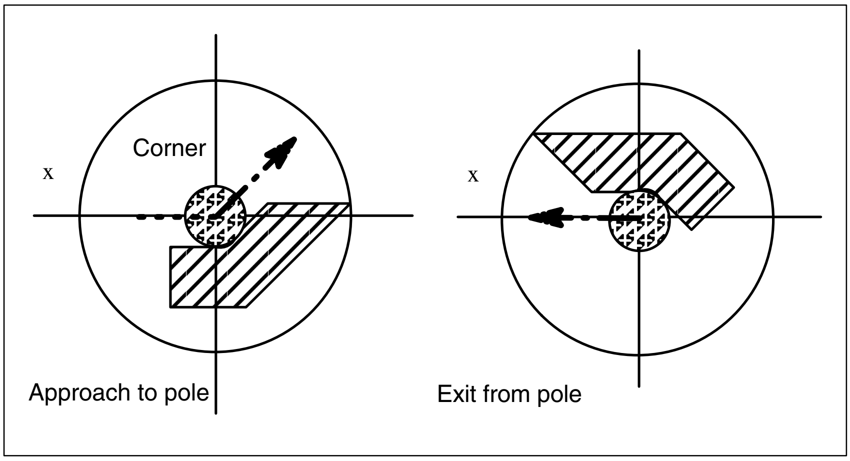



Fig. 2-8 Machining on one pole side

Requirements:
AUTOMATIC mode,
MD 24911: TRANSMIT_POLE_SIDE_FIX_1 = 1 or 2 or
MD 24951: 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 MD 24911: TRANSMIT_POLE_SIDE_FIX_1 = 0 or
MD 24951: TRANSMIT_POLE_SIDE_FIX_2 = 0 is set (pole traversal), then the **smallest possible** rotation is generated at the beginning of the block that implements exit from the pole. Depending on this rotation, the axis then traverses either in front of or behind the turning center.

When MD 24911: TRANSMIT_POLE_SIDE_FIX_1 = 1 or
MD 24951: TRANSMIT_POLE_SIDE_FIX_2 = 1 machining continues **in front of** the turning center (linear axis in positive control range);
when MD 24911: TRANSMIT_POLE_SIDE_FIX_1 = 2 or
MD 24951: TRANSMIT_POLE_SIDE_FIX_2 = 2 machining is **behind** the turning center (linear axis in the negative control range).

**Transformation selection outside pole**

The control system moves the axes involved in the transformation without evaluating machine data $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.

### 2.1.7 Working area limitations

**Initial state**

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.

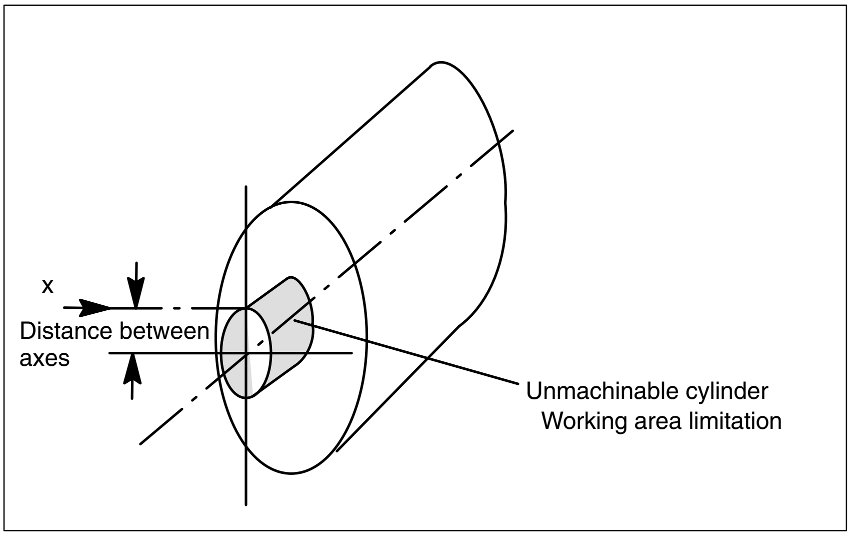



Fig. 2-9 Working area limitation based on offset linear axis

**Traversal 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 path leads past the illegal range, the control automatically reduces the feedrate and path acceleration rate to ensure that the settings of the machine axes (MD 32000: MAX_AX_VELO[AX*] and MD 32300: MAX_AX_ACCEL[AX*]) are not exceeded. The closer the path is to the working area limitation, the greater the reduction in the feedrate may be.

### 2.1.8 Overlaid movements with TRANSMIT in SW 4

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

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

### 2.1.9 Monitoring of rotary axis rotations over 360°

**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 instead. See:

**References:** /FB/, H2, “Auxiliary Function Output to PLC”
/FB/, S5, “Synchronized Actions”

In single block mode, the control processes individual blocks explicitly. In other modes, the sub-blocks are traversed with Look Ahead like a single block. A limitation of the rotary axis setting range is monitored by the software limit switch monitoring function.

### 2.1.10 Supplementary conditions

**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”. The active prevention of axis traversal through the pole implemented in SW 2 and 3 has been eliminated in SW 4.

**Several pole traversals**

A block can traverse the pole any number of times (e.g. programming of a helix with several turns). The parts 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. The relevant restrictions applying in SW 2 and 3 have been eliminated in SW 4.

**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 in SW 4.

**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**

When TRANSMIT is active, the channel identifier of posBCS[ax[3]] must have a different name to the geometry axes in the part program. 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. When TRANSMIT is active, contour information is not processed via ax[3].

**REPOS**

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

**Block search**

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

## 2.2 TRACYL

---

**Note**

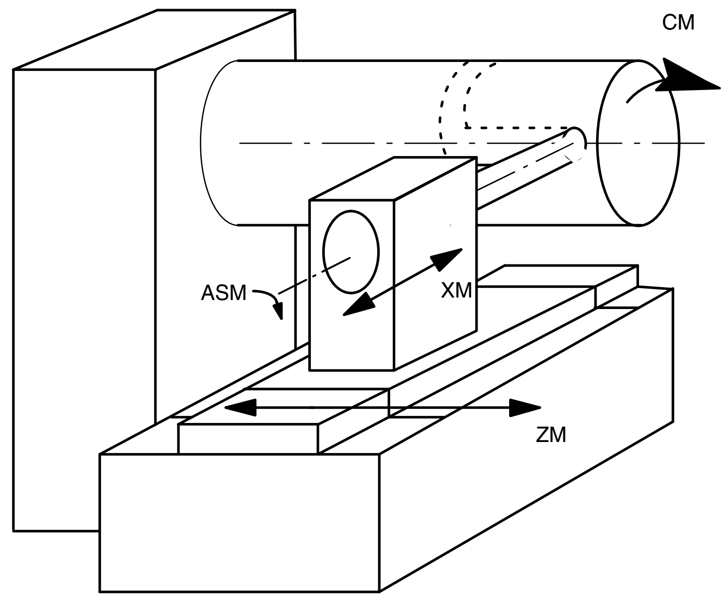
The TRACYL transformation described below requires that unique names are assigned to machine axes, channels and geometry axes when the transformation is active. Compare
MD 10000: AXCONF_MACHAX_NAME_TAB,
MD 20080: AXCONF_CHANAX_NAME_TAB,
MD 20060: AXCONF_GEOAX_NAME_TAB.
This is the only method of ensuring unique assignments.

---

**Task assignment**

Groove machining, see diagram.

**Axis configuration 1**




Legend:

XM Infeed axis, perpendicular to rotary axis
ZM Axis parallel to rotary axis
CM Rotary axis
ASM Working spindle

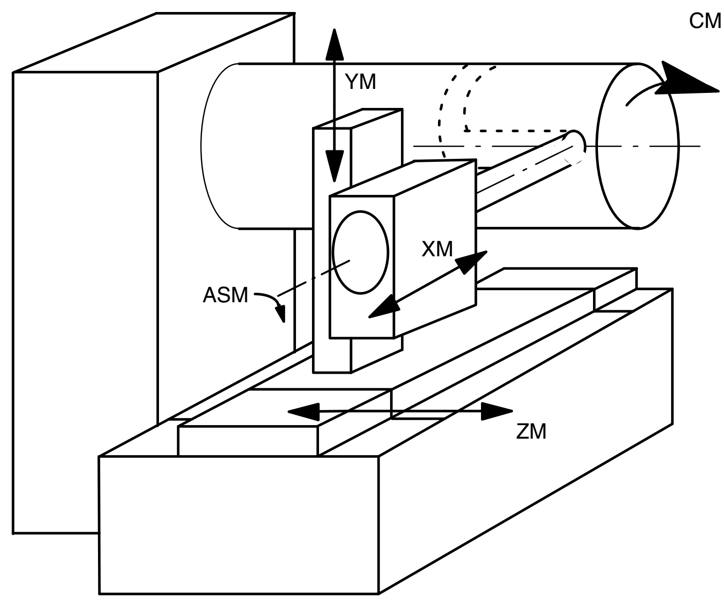
Fig. 2-10 Machining grooves on generated cylinder surface with X-C-Z kinematics

**Axis configuration 1**

The generated cylinder surface curve transformation allows a traversing path to be specified with respect to the generated surface of a cylinder coordinate system. The machine kinematics must correspond to the cylinder coordinate system. It must include one or two linear axes and a rotary axis. The two linear axes must be mutually perpendicular. The rotary axis must be aligned in parallel to one of the linear axes and intersect the second linear axis. In addition, the rotary axis must be co-linear to the cylinder coordinate system.

If there is only one linear axis (X), only grooves which are parallel to the periphery of the cylinder can be generated. In the case of two linear axes (X,Z), the groove pattern on the cylinder is optional. See Fig. 2-10.

**Axis configuration 2**




Legend:

| | |
|---|---|
| XM | Infeed axis, perpendicular to rotary axis |
| YM | Supplementary axis |
| ZM | Axis parallel to rotary axis |
| CM | Rotary axis |
| ASM | Working spindle |

Fig. 2-11 Machining grooves on generated cylinder surface with X-Y-Z-C kinematics

If a third linear axis is available (Fig. 2-11) which can produce a right-handed Cartesian coordinate system with the other two linear axes (axis configuration 1), then it is used to offset the tool **parallel to the programmed path** by means of tool radius compensation. thereby allowing grooves with rectangular traversing section to be generated.

**Functionality**

During transformation (both axis configurations), the full functionality of the control is available, both for processing from the NC program and in JOG mode (see 2.2.6).

**Groove cross-section**

In the case of axis configuration 1, longitudinal grooves along the rotary axis are subject to parallel limits only if the groove width corresponds exactly to the tool radius.

Grooves in parallel to the periphery (transverse grooves) are not parallel at the beginning and end.

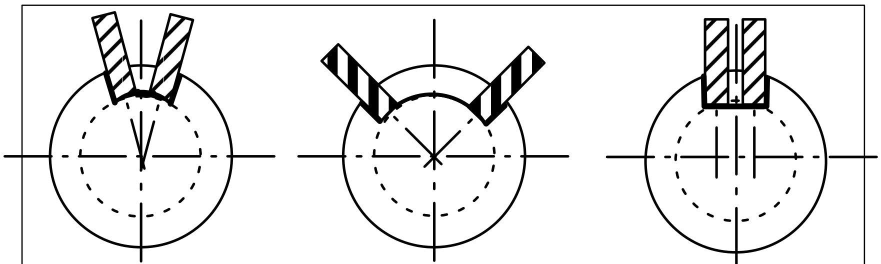



Fig. 2-12 Grooves with and without groove wall offset

## 2.2.1 Preconditions for TRACYL

**Number of transformations**

Up to 10 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 and10. The first machine data has the same meaning as described for TRANSMIT:

$MC_TRAFO_GEOAX_ASSIGN_TAB_n
$MC_TRAFO_TYPE_n
$MC_TRAFO_AXES_IN_n.

The special settings described below apply to $MC_TRAFO_TYPE_n and $MC_TRAFO_AXES_IN_n with respect to generated cylinder surface transformation (TRACYL).

**Number of TRACYL structures**

Three of the 10 permitted data structures for transformations may be assigned to the TRACYL function. They are characterized by the fact that the value assigned with $MC_TRAFO_TYPE_n is 512 or 513 or 514.
For these 3 TRACYL transformations, the following machine data must be set in a defined way:

$MC_TRACYL_ROT_AX_OFFSET_t
$MC_TRACYL_ROT_SIGN_IS_PLUS_t
$MC_TRACYL_BASE_TOOL_t

In this case, t specifies the number of the declared TRACYL transformation (maximum of 3).

**Axis configuration**

The following overview shows the relationship between the axes of the machine illustrated in Fig. 2-11 and the relevant axis data.

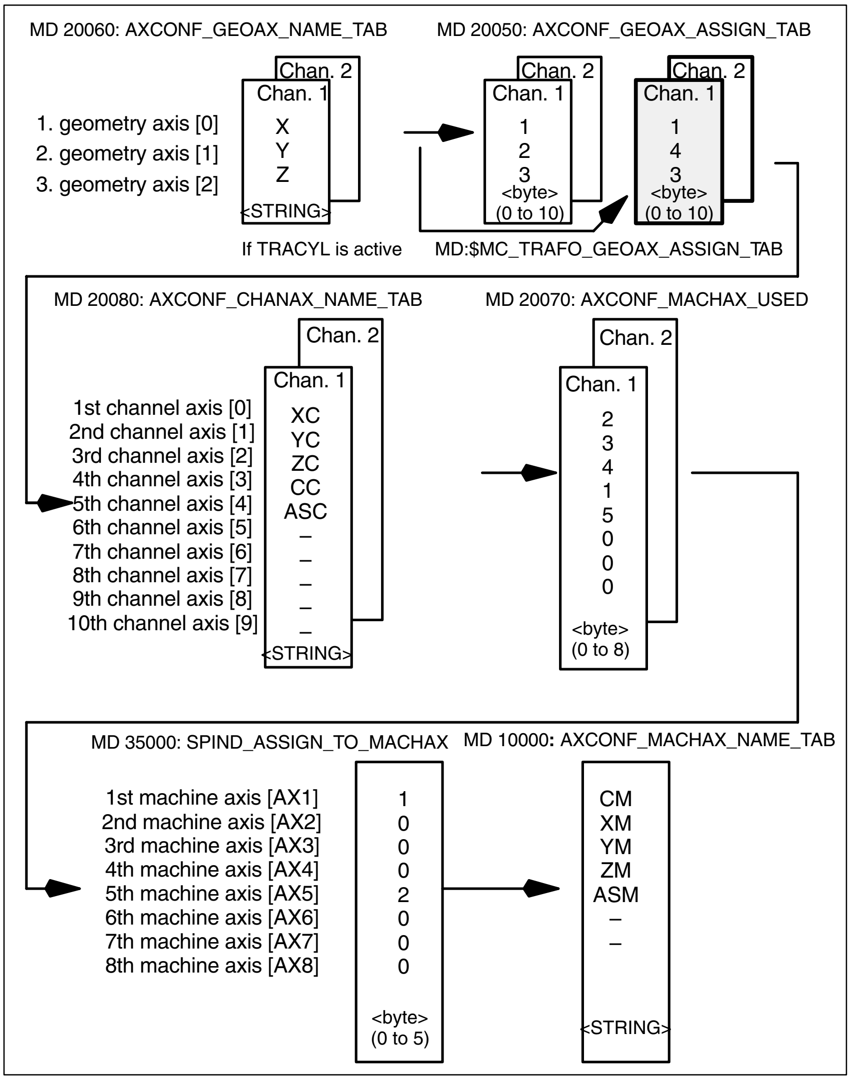



Fig. 2-13 Axis configuration for the example in Fig. 2-11

The configurations highlighted in Fig. 2-13 apply when TRACYL is active.

**Assignment of names to geometry axes**

According to the above axis configuration overview, the geometry axes to be involved in the TRACYL operation must be defined with:

$MC_AXCONF_GEOAX_NAME_TAB[0]="X"
" _TAB[1]="Y"
" _TAB[2]="Z"
(name selection according to Fig. 2-13, also corresponds to default setting).

**Assignment of geometry axes to channel axes**

These assignments are made depending on whether or not TRACYL is active.

– TRACYL not active
A Y axis is traversed normally.

```
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]=1
                    ”           _TAB[1]=2
                    ”           _TAB[2]=3
```

– TRACYL active
The Y axis becomes the axis in the peripheral surface direction of the cylinder coordinate system.

```
$MC_TRAFO_GEOAX_ASSIGN_TAB[0]=1
                    ”         _TAB[1]=4
                    ”         _TAB[2]=3
```

**Entry of channel axes**

Axes which do not belong to the Cartesian coordinate system are entered.

```
$MC_AXCONF_CHANAX_NAME_TAB[0]=“XC”
                ”             _TAB[1]=“YC”
                ”             _TAB[2]=“ZC”
                ”             _TAB[3]=“CC”
                ”             _TAB[4]=“ASC”
```

**Assignment of channel axes to machine axes**

With the cd of the channel axes as a reference, the machine axis number to which the channel axes have been assigned is transferred to the control system.

```
$MC_AXCONF_MACHAX_USED[0]=2
                ”             [1]=3
                ”             [2]=4
                ”             [3]=1
                ”             [4]=5
```

(entries corresponding to Fig. 2-11)

**Identification of spindles**

The user defines whether each machine axis is a spindle (value > 0: spindle number) or a path axis (value 0).

```
$MA_SPIND_ASSIGN_TO_MACHAX[0]=1
                ”                [1]=0
                ”                [2]=0
                ”                [3]=0
                ”                [4]=2
```

**Assignment of names to machine axes**

With the cd of the machine axes as a reference, a machine axis name is transferred to the control.

```
$MN_AXCONF_MACHAX_NAME_TAB[0]=“CM”
                ”                 [1]=“XM”
                ”                 [2]=“YM”
                ”                 [3]=“ZM”
                ”                 [4]=“ASM”
```

## 2.2.2 TRACYL-specific settings

**Type of transformation**

The following paragraph describes how the transformation type is specified.

**TRAFO_TYPE_n**

The user must specify the transformation type for the transformation data blocks (maximum n = 10). 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 no groove side offset with supplementary linear axis. Transformation type 514 can also be activated with groove side offset by means of an additional parameter. See also Subsection 2.2.3 Activation.

| | |
|---|---|
| Example of VALUE 512: | MD 24100: TRAFO_TYPE_1=512 |

The setting must be made before TRACYL(d,t) is called. “t” is the number of the declared TRACYL transformation.

The TRACYL transformation requires only a rotary axis and a linear axis positioned perpendicular to the rotary axis. A real Y axis is used with transformation type 514 in order, for example, to compensate a tool offset.

**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 user memory of the second linear axis is small and will not be used to execute the part program.

The existing settings for $MC_TRAFO_GEOAX_ASSIGN_TAB_n apply.

**Grooves with groove side offset**

The tool offset is included as required in the calculation for the TRACYL transformation with groove side offset.

**Axis image**

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

**TRAFO_AXES _IN_n**

Three (or 4) channel axis numbers must be specified for the transformation data block n:

| | |
|---|---|
| $MC_TRAFO_AXES_IN_1[0]= | Channel axis number of axis radial to rotary axis. |
| $MC_TRAFO_AXES_IN_1[1]= | Channel axis number of rotary axis |
| $MC_TRAFO_AXES_IN_1[2]= | Channel axis number of axis parallel to rotary axis |
| $MC_TRAFO_AXES_IN_1[3]= | Channel axis number of additional axis, parallel to generated cylinder surface and perpendicular to rotary axis (if axis configuration 2 is used) |

**Example based on Fig. 2-11:**
$MC_TRAFO_AXES_IN_1[0]=1
$MC_TRAFO_AXES_IN_1[1]=4
$MC_TRAFO_AXES_IN_1[2]=3
$MC_TRAFO_AXES_IN_1[3]=2

The setting must be made before TRACYL(d) or TRACYL(d,t) is called. The axis numbers must refer to the channel axis sequences defined with $MC_TRAFO_GEOAX_ASSIGN_TAB_n.

**Grooves without groove side offset**

**For transformation type 514**, the following index assignments for $MC_TRAFO_AXES_IN_n[ ] must be selected.
Meaning of indices in relation to basic 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:

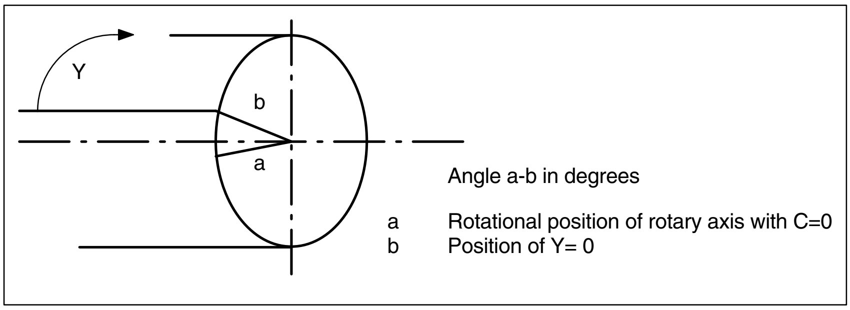



Fig. 2-14 Center of rotation of axis on generated cylinder surface

**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:

$MC_TRACYL_ROT_AX_OFFSET_t= ... ; degrees

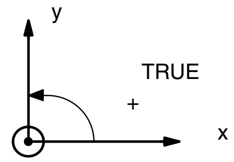
In this case, “t” is substituted by the number of the TRACYL transformations declared in the transformation data blocks
(t must not be more than 2).

**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 counter-clockwise when viewed against the z axis, then the machine data must be set to TRUE, otherwise to FALSE.




$MC_TRACYL_ROT_SIGN_IS_PLUS_t=TRUE

In this case, “t” is substituted by the number of the TRACYL transformations declared in the transformation data blocks
(t must not be more 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.

- MD 22534: 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:** /FB/, K2, “Coordinate Systems, Axis Types, Axis Configurations, Actual-Value System for Workpiece, External Zero Offset”

**Position of tool zero**

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

**TRACYL_ BASE_TOOL_t**

Machine data $MC_TRACYL_BASE_TOOL_t is used to inform the control of the position of the tool zero point 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.

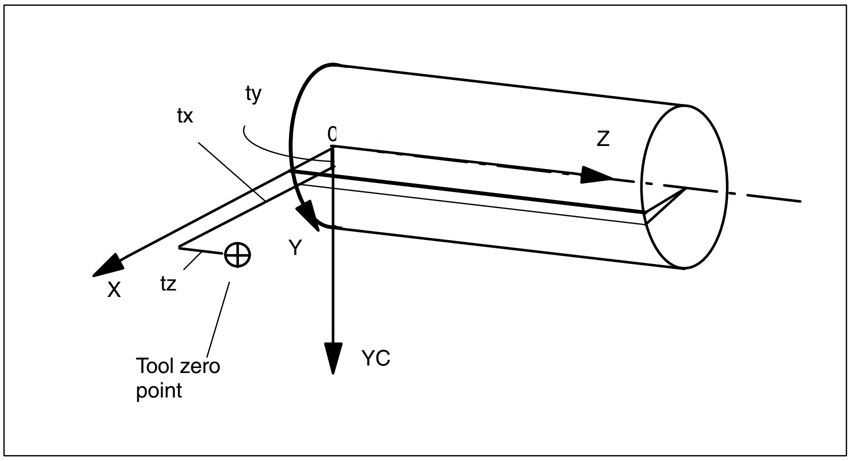



Fig. 2-15 Position of tool zero in relation to machine zero (see Fig. 2-11)

Example:
$MC_TRACYL_BASE_TOOL_t[0]=tx
” [1]=ty
” [2]=tz

In this case, t is replaced by the number of the TRACYL transformations declared in the transformation data blocks.
(t must not be more than 2).

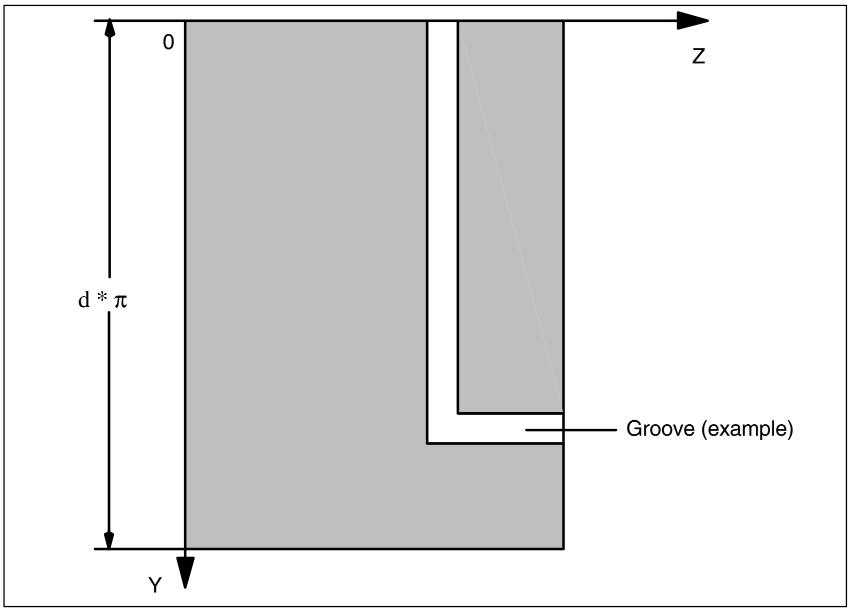



Fig. 2-16 Cylinder coordinate system for Fig. 2-15

## 2.2.3 Activation of TRACYL

**TRACYL**

After the settings described above have been made, the TRACYL function can be activated:

TRACYL(d)
or
TRACYL(d,t) TRACYL(reference diameter, Tracyl data block)

The first declared TRACYL function is activated with TRACYL(d). TRACYL(t) activates the t-th declared TRACYL function. t may not be more than 2. The value d stands for the current diameter of the cylinder to be machined.

Between activation of the function and deactivation as described below, the traversing movements for the axes of the cylinder coordinate system can be programmed.

**Transformation type 514 with groove side offset**

An additional call parameter is used for transformation type 514; this is the third parameter with which TRACYL transformation with groove side offset can be selected:

TRACYL(reference diameter, Tracyl data block, groove side offset).

- Reference diameter: Obligatory parameter (must always be defined) Value range: >0
- Tracyl data block: Optional parameter, preset value is 1
  Value range: 1,2
- Groove side offset: Optional parameter, preset value corresponds to value specified in machine data
  MD 24808: TRACYL_DEFAULT_MODE_1) or
  MD 24858: TRACYL_DEFAULT_MODE_2)
  Value range: 0,1

## 2.2.4 Deactivation of TRACYL function

**TRAFOOF**

Keyword TRAFOOF deactivates an active transformation. When the transformation is deactivated, the base coordinate system is again identical to the machine coordinate system.
An active TRACYL transformation is likewise deactivated if one of the other transformations is activated in the relevant channel.
(e.g. TRANSMIT, TRAANG, TRAORI).
**References:** /FB/, F2, “5-Axis Transformation”

## 2.2.5 Special system reactions with TRACYL

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

**Please note on selection**

- An intermediate motion block is not inserted (phases/radii).
- A spline block sequence must be terminated.
- Tool radius compensation must be deselected.
- The frame which was active prior to TRACYL is deselected by the control (corresponds to "Reset programmed frame" G500).
- An active working area limitation is deselected by the control for the axes affected by the transformation
  (corresponds to programmed WALIMOF).
- Continuous path control and rounding are interrupted.
- DRF offsets must have been deleted by the operator.
- In the case of cylinder generated surface curve transformation with groove wall compensation (axis configuration 2, TRAFO_TYPE_n=513), the axis used for the correction (TRAFO_AXES_IN_n[3]) must be set to zero (y=0) so that the groove is machined in the center of the programmed groove center line.

**Please note on deselection**

The same points apply as for selection.

**Restrictions when TRACYL is active**

The restrictions listed below must be noted when the TRACYL function is active:

**Tool change**

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

**Supplementary conditions for TRACYL without groove side offset**

When TRANSMIT is active, the channel identifier of posBCS[ax[3]] must have a different name to the geometry axes in the part program. If posBCS[ax[3]] is written only outside the TRACYL transformation, this restriction does not apply if the axis has been assigned to a geometry axis. When TRACYL is active, contour information is not processed via ax[3].

**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.

**Rotary axis**

The rotary axis cannot be programmed because it is occupied by a geometry axis and cannot thus be programmed directly as a channel axis.

**Extensions with SW 6.4 and later**
An offset in the rotary axis CM can be entered, for example, by compensating the inclined position of a workpiece in a frame within the frame chain. The x and y values are then as illustrated in the following diagram.

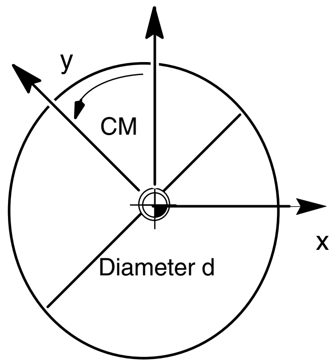



Fig. 2-17 Rotary axis offset with TRACYL

With SW 6.4 and later, this offset can also be included in the transformation as an offset in the rotary axis or as a y offset. To ensure that the total axial frame of the Tracyl rotary axis, i.e. the translation, fine offset, mirroring and scaling, is included in the transformation, the following settings must be made:

MD 24805: TRACYL_ROT_AX_FRAME_1 = 1
MD 24855: TRACYL_ROT_AX_FRAME_2 = 1

---

**Note**

Changes in the axis assignments are converted every time the transformation is selected or deselected. For further information about axial offsets for rotary axes to the SZS as an offset on the peripheral surface, please see:

**References:** /FB/, K2, “Coordinate Systems, Frames”

---

**Axis utilization**

The axes:

– in the generated cylinder surface perpendicular to the rotary axis (Y) and

– additional axis (YC)

may not be used as a positioning or oscillation axis.

**Exceptions**

Axes affected by the transformation cannot be used:

- as a preset axis (alarm)
- to approach the fixed point (alarm)
- for referencing (alarm)

**Interruption of parts program**

The following points must be noted with respect to interrupting parts program processing in connection with TRACYL:

**Automatic after Jog**

If parts program processing is interrupted when the transformation is active followed by traversal in Jog mode, then the following must be noted when Automatic is selected again:

- The transformation is active in the approach block from the current position to the point of interruption. No monitoring for collisions takes place.

---

**Warning**

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

---

**START after RESET**

If parts program processing is aborted with RESET and restarted with START, then the following must be noted:

- The remaining parts program is traversed reproducibly only if all axes are traversed to a defined position by means of a linear block (G0 or G1) at the beginning of the parts program. A tool which was active on RESET may no longer be taken into account by the control (settable via machine data).

## 2.2.6 Jog

**Special features relating to JOG**

When generated cylinder surface transformation with groove wall 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. When groove wall compensation is active, the axes movement therefore differs from the situation when the correction function is deselected. The parts program can therefore be continued (REPOS) after a parts program interruption.

## 2.3 TRAANG (inclined axis)

---

**Note**

The TRAANG transformation described below requires that unique names are assigned to machine axes, channels and geometry axes when the transformation is active. Compare
MD 10000: AXCONF_MACHAX_NAME_TAB,
MD 20080: AXCONF_CHANAX_NAME_TAB,
MD 20060: AXCONF_GEOAX_NAME_TAB.
This is the only method of ensuring unique assignments.

---

**Task assignment**

Grinding operations

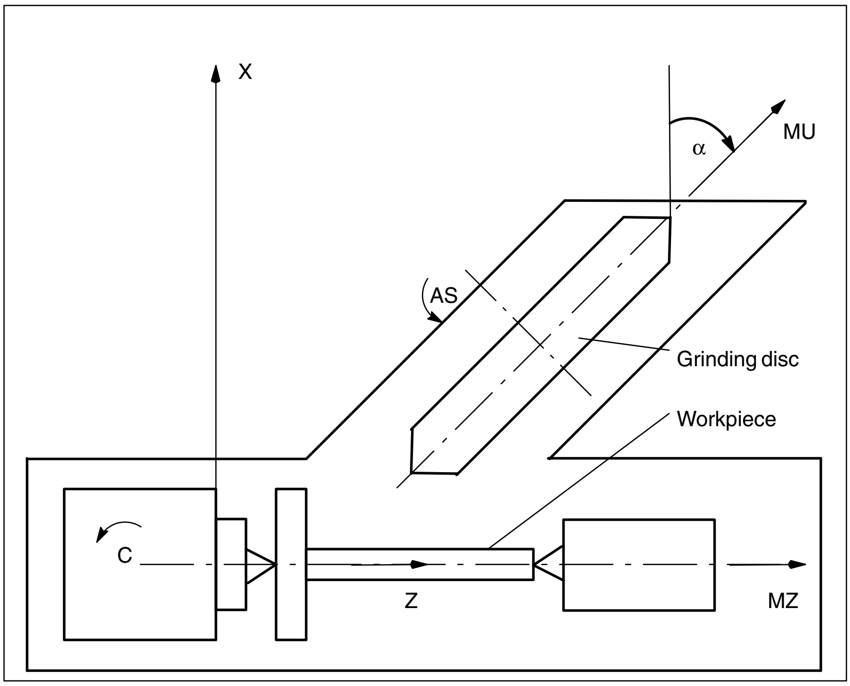



Fig. 2-18 Machine with inclined infeed axis

Caption:

| | |
|---|---|
| X, Z | Cartesian coordinate system for programming |
| C | Rotary axis |
| AS | Working spindle |
| MZ | Machine axis (linear) |
| MU | Inclined axis |

The following range of machining operations is available:

1. Longitudinal grinding
2. Face grinding
3. Grinding of a specific contour
4. Oblique plunge-cut grinding

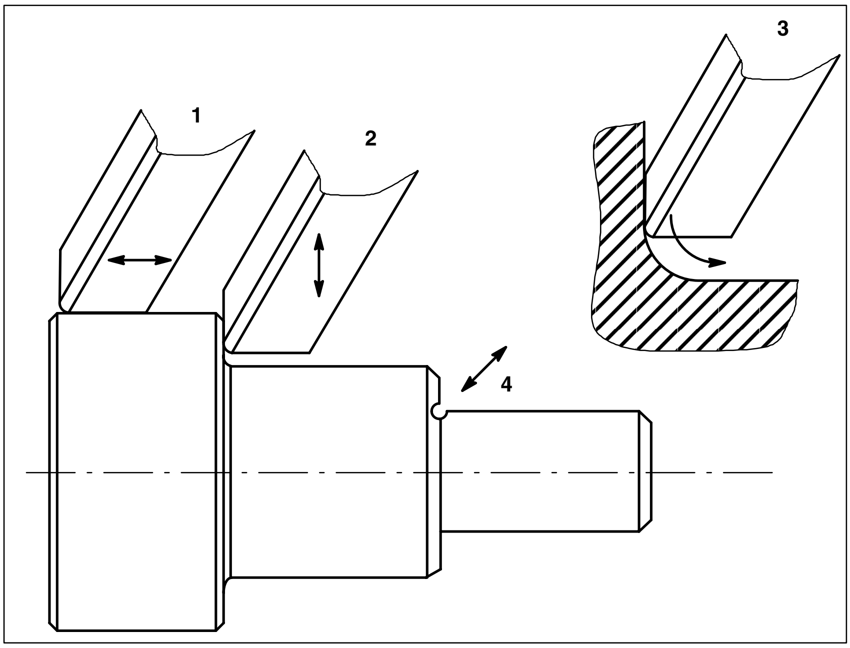



Fig. 2-19 Possible grinding operations

## 2.3.1 Preconditions for TRAANG (inclined axis)

**Axis configuration**

To be able to program in the Cartesian coordinate system (see Fig. 2-18: X, Y, Z), it is necessary to inform the control of the correlation between this coordinate system and the actually existing machine axes (MU,MZ):

- Assignment of names to geometry axes
- Assignment of geometry axes to channel axes
  - general situation (inclined axis not active)
  - inclined axis active
- Assignment of channel axes to machine axis numbers
- Identification of spindles
- Allocation of machine axis names

With the exception of the "– Inclined axis active" point, the procedure is the same as for the normal axis configuration.

**References:** /FB/, K2, "Coordinate Systems, Axis Types, Axis Configurations, Actual-Value System for Workpiece, External Zero Offset"

**Number of transformations**

Up to 10 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. The following sections include descriptions of these data:

$MC_TRAFO_GEOAX_ASSIGN_TAB_n
$MC_TRAFO_TYPE_n
$MC_TRAFO_AXES_IN_n

**Number of inclined axes**

Two of the 10 permitted data structures for transformations may be assigned to the inclined axis function. They are characterized by the fact that the value assigned with $MC_TRAFO_TYPE_n is 1024.

**Axis configuration**

The axes of the grinding machine illustrated in Fig. 2-18 must be entered as follows in the machine data:

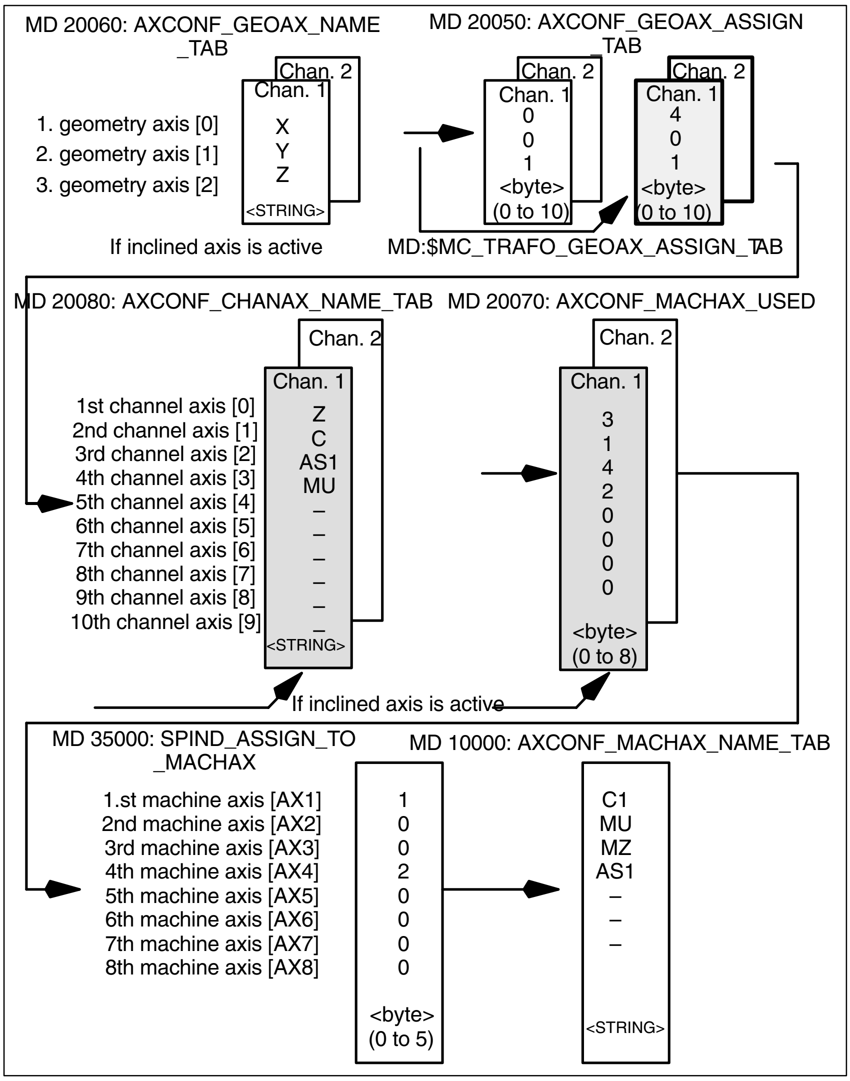



Fig. 2-20 Axis configuration for the example in Fig. 2-18

The highlighted configuration in Fig. 2-20 applies when TRAANG is active.

## 2.3.2 TRAANG-specific settings

**Type of transformation**

**TRAFO_TYPE_n**

The user must specify the transformation type in machine data $MC_TRAFO_TYPE_n for the transformation data blocks (maximum n = 10). The value for an inclined axis is 1024:
MD 24100: TRAFO_TYPE_1=1024

**Axis image**

**TRAFO_AXES_IN_n**

Two channel axis numbers must be specified for the transformation data block n:

MD 24110: TRAFO_AXES_IN_1[0] = 4 ; channel axis number of inclined axis
MD 24110: TRAFO_AXES_IN_1[1] = 1 ; channel axis number of parallel axis

for Z
MD 24110: TRAFO_AXES_IN_1[2]=0 ; channel axis number not active

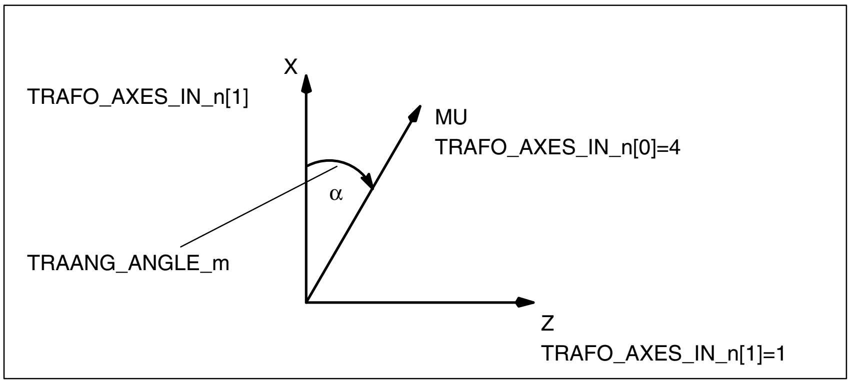



Fig. 2-21 Parameter TRAANG_ANGLE_m

**Assignment of geometry axes to channel axes**

Example:

MD 24430: TRAFO_TYPE_5 = 8192 Chaining

MD 24110: TRAFO_AXIS_IN_1[0..x]

MD 24434: TRAFO_GEOAX_ASSIGN_TAB_5[0] = 1 Definition geo axis assignment of transf. 1
MD 24434: TRAFO_GEOAX_ASSIGN_TAB_5[1] = 6 Definition geo axis assignment of transf. 1
MD 24434: TRAFO_GEOAX_ASSIGN_TAB_5[2] = 3 Definition geo axis assignment of transf. 1

MD 24996: TRACON_CHAIN_2[0] = 2 Input variables in TRACON
MD 24996: TRACON_CHAIN_2[1] = 3 Input variables in TRACON
MD 24996: TRACON_CHAIN_2[2] = 0 Input variables in TRACON
MD 24996: TRACON_CHAIN_2[3] = 0 Input variables in TRACON

**Angle of inclined axis**

**TRAANG_ ANGLEm**

Machine data $MC_TRAANG_ANGLE_m is used to inform the control of the angle which exists between a machine axis and the inclined axis in degrees.

$MC_TRAANG_ANGLE_m = Angle between a
Cartesian axis and the associated inclined machine axis in degrees. The angle is counted positively in the clockwise direction (see Fig. 2-18, angle $\alpha$).

In this case, “m” is substituted by the number of the TRAANG transformation declared in the transformation data blocks.
m must not be more than 2.

**Permissible angular range**

The permissible angular range is:
–90° < TRAANG_ANGLE_m < 0°
0° < TRAANG_ANGLE_m < 90°

No transformation is required for 0°.
With +/ – 90° the inclined axis is positioned parallel to the second linear axis.

**Position of tool zero point**

**TRAANG_ BASE_TOOL_m**

Machine data $MC_TRAANG_BASE_TOOL_m is used to inform the control of the position of the tool zero point in relation to the origin of the coordinate system declared for the inclined axis function. The machine data has three components for the three axes of the Cartesian coordinate system.

Zero is entered as default.
The corrections are not converted when the angle is changed.

**Optimization of velocity control**

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

**TRAANG_PARALLEL_VELO_RES_m**

Machine data $MC_TRAANG_PARALLEL_VELO_RES_m is used to set the velocity reserve which is held ready on the parallel axis (see $MC_TRAFO_AXES_IN_n[1]) for the compensatory motion.

Value range: 0 ... 1

**0** When value 0 is set, the control automatically determines the reserve: the axes are limited with equal priority.
(= default setting)

**>0** With values of >0, the reserve is fixed at

| | | |
|---|---|---|
| $MC_TRAANG_PARALLEL _VELO_RES_m | * | Permissible machine axis velocity value of parallel axis |

The velocity characteristics of the vertical axis are determined by the control on the basis of the reserve.

**TRAANG_PARALLEL_ACCEL_RES_m**

Machine data $MC_TRAANG_PARALLEL_ACCEL_RES_m is used to set the axis acceleration reserve which is held ready on the parallel axis (see $MC_TRAFO_AXES_IN_n[1]) for the compensatory motion.

Value range: 0 ... 1

**0** When value 0 is set, the control automatically determines the reserve: the axes are accelerated with equal priority.
(= default setting)

**>0** With values of >0, the acceleration rate is fixed at

| | | |
|---|---|---|
| $MC_TRAANG_PARALLEL _ACCEL_RES_m | * | Permissible machine axis velocity value of the parallel axis |

The velocity characteristics of the vertical axis are determined by the control on the basis of the reserve.

**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.

- MD 22534: TRAFO_CHANGE_M_CODE

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

---

**Note**

No M code is output if the machine data is set to one of the values 0 to 6, 17 or 30.

---

**References:** /FB/, K2, “Coordinate Systems, Axis Types, Axis Configurations, Actual-Value System for Workpiece, External Zero Offset”

## 2.3.3 Activation of TRAANG

**TRAANG(a)**

After the settings described above have been made, the TRAANG function can be activated:

TRAANG(a) or
TRAANG(a,n)

The first declared "inclined axis" transformation is activated with TRAANG(a). The angle of the inclined axis can be specified with "a".

**Software versions < 6.4**
If angle "a" is omitted or zero is entered, the transformation is activated with the parameterization of the previous selection.
The default selection according to the machine data applies for the initial selection.

**SW 6.4 and higher**

If a (angle) is omitted (e.g. TRAANG(), TRAANG(,n) ), the transformation is activated with the parameter settings of the previous selection. On the first selection, the presettings according to the machine data apply. An angle a = 0 (e.g. TRAANG(0), TRAANG(0,n)) is a valid parameter setting and is no longer equivalent to the omission of the parameter, as in the case of older versions. The permissible value range for a is: –90 degrees < a < + 90 degrees.

TRAANG(a,n) activates the nth declared inclined axis transformation.
This form is required only if several transformations are activated in the channel.
n must not be more than 2.

**Programming variants**

TRAANG(a,1) == TRAANG(a,0) == TRAANG(a,) == TRAANG(a)

Between activation of the function and deactivation as described below, the traversing movements for the axes of the Cartesian coordinate system must be programmed.

## 2.3.4 Deactivation of TRAANG

**TRAFOOF**

Keyword TRAFOOF deactivates an active transformation. When the transformation is deactivated, the base coordinate system is again identical to the machine coordinate system.
An active TRAANG transformation is likewise deactivated if one of the other transformations (e.g. TRACYL, TRANSMIT, TRAORI) is activated in the appropriate channel
(e.g. TRACYL, TRANSMIT, TRAORI).

**References:** /FB/, F2, "5-Axis Transformation"

### 2.3.5 Special system reactions with TRAANG

The transformation can be selected and deselected via parts 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 $MC_TRAFO_AXES_IN_n must be synchronized on a block-related basis (e.g. no traversing instruction with POSA...).

**Restrictions**

**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 with SW 6.4 and later**
When TRAANG is selected and deselected, the assignment between geometry axes and channel axes can change. With SW 6.4 and later, 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 further information about these frame offsets with transformations, see:
**References:** /FB/, K2, “Axes, Coordinate Systems, Frames”

**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, with oblique machining operations,

## 2.3.6 Programming an inclined axis: G05, G07 (from SW 5.3)

**Function**

The following functions are available:

- Position programming and display in the Cartesian coordinate system
- Cartesian calculation of tool offset and zero offset
- Programming of angles for the inclined axis in the NC program
- Approach starting position for inclined plunge cutting (G07)
- Inclined plunge cutting (G05)
- The grinding wheel can be moved in JOG mode either according to Cartesian coordinates or in the direction of the inclined axis (display remains Cartesian).
  The selection is made via DB21-28 DBX29.4 “PTP travel”. If PTP travel is activated, only the real U axis moves, the Z axis display is updated.

**Programming**

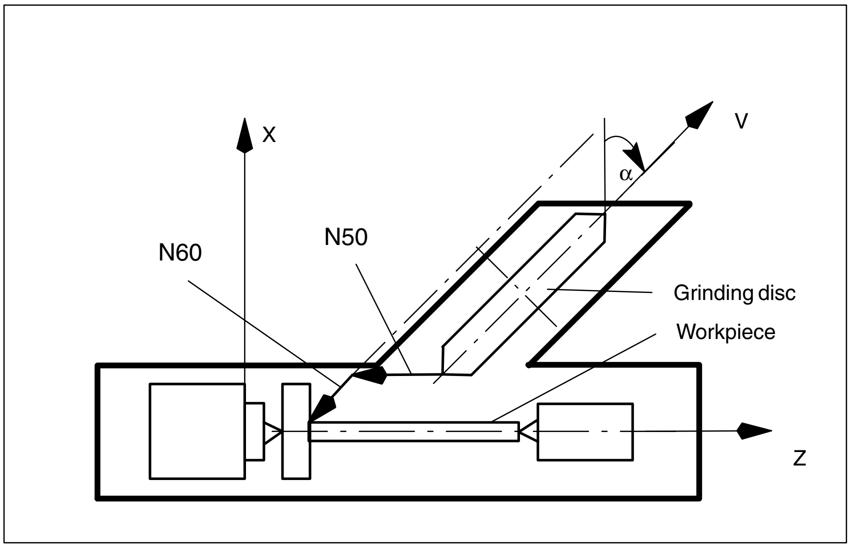



Fig. 2-22 Machine with inclined infeed axis

Example:

```
N...                      ; Program angle for the inclined axis
N50 G07 X70 Z40 F4000     ; Approach starting position
N60 G05 X70 F100          ; Inclined plunge cutting
N...
```

**Supplementary conditions**

- It is only meaningful to select the function “Cartesian PTP travel” in JOG mode (motion according to G05) if transformation is active (TRAANG). Note the value set in MD 20140 TRAFO_RESET_VALUE.
- REPOS offsets must be traversed back in JOG mode in the Cartesian coordinates while “PTP travel” is not active.
- The Cartesian working area limitation is monitored for overtravel in jog mode if “PTP travel” is active and the axis will brake before overtraveling. If “PTP travel” is not active, the axis can be traversed right up to the working area limitation.

For further information, see Section “Cartesian PTP travel”.

## 2.4 Chained transformations

**Introduction**

SW 5 and higher supports chaining of the kinematic transformations described here:

- TRANSMIT
- TRACYL
- TRAANG (inclined axis)

as well as those described in

**References:** /FB/, F2, “3 to 5-axis transformations”

- Orientation transformations
- Universal milling head

with another transformation of the “Inclined axis” type.

**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 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 MD:
MD 10000: AXCONF_MACHAX_NAME_TAB,
MD 20080: AXCONF_CHANAX_NAME_TAB,
MD 20060: AXCONF_GEOAX_NAME_TAB.
This is the only method of ensuring unique assignments.

---

**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 the channel axes
  – General case (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 each single transformation **and** for each **chained** transformation)
  – Transformation type
  – Axes included in the transformation
  – Assignment of geometry axes to the channel axes with active transformation
  – According to transformation, also
  rotational position of the coordinate system,
  direction of rotation,
  tool zero or original coordinate system
  angle of 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 and10.

**Number of chained transformations**

Of the maximum of 10 transformations for a channel, up to **two chained** transformations can 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.

## 2.4.1 Activating chained transformations

**TRACON**

A chained transformation is activated via:

**TRACON**(trf, par)

where:

| | |
|---|---|
| trf | Number of the chained transformation:<br>0 or 1 for first/single chained transformation.<br>If nothing is programmed here, then this has the same meaning as specifying value 0 or 1,<br>i.e. the first/single transformation is activated.<br>2 for the second chained transformation.<br>(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.<br>For example, the angle of the inclined axis. If parameters are not set, the defaults or the parameters last used take effect.<br>Commas must be used to ensure that the specified parameters are evaluated in the sequence in which they are expected, if defaults are to act for previous parameters.<br>In particular, a comma is required before at least one parameter, even though it is not necessary to specify trf. For example:<br>TRACON( , 3.7). |

If another transformation was previously activated, it is implicitly disabled by means of TRACON().

## 2.4.2 Switching off chained transformations

**TRAFOOF**

A chained transformation is switched off with TRAFOOF just like any other transformation.

## 2.4.3 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**

Section 6.3 contains configuration examples for single transformations and the transformation chains created from them.

## 2.4.4 Persistent transformation

**Function**

A persistent transformation is active continuously and acts in relation to 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 further information about programming, see
**References:** /PGA/, Transformations, "Chained transformation"

**Selection and deselection**

The persistent transformation is selected by means of the machine data below:

| | |
|---|---|
| MD 20144:TRAFO_MODE_MASK, | |
| Bit 0 = 1 | MD 20144 TRAFO_RESET_VALUE defines persistent transformation. |
| MD 20140:TRAFO_RESET_VALVUE= | Number of transformation data block of persistent transformation |
| These MD must also be set: | i.e. taken into account: |
| MD 20110:RESET_MODE_MASK | |
| Bit 0 = 1, | Bit 7 |
| Bit 7 = 0. | MD 20140:TRAFO_RESET_VALUE |
| MD 20112:START_MODE_MASK | MD 20140:TRAFO_RESET_VALUE |
| MD 20118:GEOAX_CHANGE_RESET= TRUE | i.e. geometry axes are reset. |

Alarm 14404 is generated if these additional data are not parametereizd correctly.

TRAFOOF deselects the active TRACON and automatically selects the persistent transformation.

**Effects on HMI operation**

As a selected persistent transformation means that one transformation is always active, the new version of the HMI operator interface has been adapted with respect to selection/deselection of transformations:

TRACON on HMI
The HMI operator interface does **not** now display TRACON, but the 1st transformation chained with 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.

TRAFOOF on HMI
**No** transformation is displayed for program instruction TRAFOOF in the G code list on the HMI operator interface, but, instead, the 1st transformation chained with TRACON, e.g. TRANSMIT. 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 actively chained transformations.

| Description | NCK variables |
|---|---|
| **No** transformation active: 0<br>**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<br>**One** transformation active: Type of 2nd chained transformation with TRACON | $P_TRAFO_CHAIN[1]<br>$AC_TRAFO_CHAIN[1] |
| Are not used unless more than 2 transformations are chained.<br>These variables **always return 0 in the current version** | $P_TRAFO_CHAIN[2]<br>$AC_TRAFO_CHAIN[2] and<br>$P_TRAFO_CHAIN[3]<br>$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**

When TRACON is selected or deselected, the frame is adjusted as if only the 1st chained transformation existed. Transformations on the virtual axis cease to be effective when TRAANG is selected.

**JOG**

The persistent transformation remains operative when axes are traversed in JOG mode.

**Supplementary conditions**

The persistent transformation does not change the basic operational 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**

On a turning machine with an additional, inclined Y axis, the transformation of the inclined axis must be a component in the machine configuration and, as such, can be otherwise ignored by the programmer. TRACYL or TRANSMIT is used to select transformations which must then include TRAANG. When the programmed transformations are deactivated, TRAANG is automatically activated again, TRACYL or TRANSMIT is displayed accordingly on the HMI operator interface.

**Machine data** for a turning machine with Y1 axis, inclined in relation to X1 but perpendicular to Z1.

```
CANDATA (1)

                    ; Kinematic without transformations

MD 20080: AXCONF_CHANAX_NAME_TAB[1] = "Y2"

MD 20050: AXCONF_GEOAX_ASSIGN_TAB[0] = 1
MD 20050: AXCONF_GEOAX_ASSIGN_TAB[1] = 0
MD 20050: AXCONF_GEOAX_ASSIGN_TAB[2] = 3

                    ; Data for TRAANG

MD 24100: TRAFO_TYP_1 = 1024 ; TRAANG Y1 axis inclined in relation to X1,
                             ; perpendicular to Z1
MD 24110: TRAFO_AXES_IN_1[0] = 2
MD 24110: TRAFO_AXES_IN_1[1] = 1
MD 24110: TRAFO_AXES_IN_1[2] = 3
MD 24110: TRAFO_AXES_IN_1[3] = 0
MD 24110: TRAFO_AXES_IN_1[4] = 0
MD 24120: TRAFO_GEOAX_ASSIGN_TAB_1[0] = 1
MD 24120: TRAFO_GEOAX_ASSIGN_TAB_1[1] = 2
MD 24120: TRAFO_GEOAX_ASSIGN_TAB_1[2] = 3
MD 24700: TRAANG_ANGLE_1 = 60
MD 24720: TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD 24721: TRAANG_PARALLEL_ACCEL_RES_1 = 0.2

                    ; Definition of persistent transformation

MD 20144:TRAFO_MODE_MASK = 1
MD 20140:TRAFO_RESET_VALVUE = 1
MD 20110:RESET_MODE_MASK = 'H01'
MD 20112:START_MODE_MASK = 'H80'
MD 20140:TRAFO_RESET_VALUE
MD 20118:GEOAX_CHANGE_RESET = TRUE

                    ; Data for TRANSMIT, TRACYL

MD 24911: TRANSMIT_POLE_SIDE_FIX_1 = 1 ; also 2, causes alarm 21617
MD 24200: TRAFO_TYP_2 = 257
MD 24210: TRAFO_AXES_IN_2[0] = 1
MD 24210: TRAFO_AXES_IN_2[1] = 4
MD 24210: TRAFO_AXES_IN_2[2] = 3
MD 24210: TRAFO_AXES_IN_2[3] = 0
MD 24210: TRAFO_AXES_IN_2[4] = 0
MD 24220: TRAFO_GEOAX_ASSIGN_TAB_2[0] = 1
MD 24220: TRAFO_GEOAX_ASSIGN_TAB_2[1] = 4
MD 24220: TRAFO_GEOAX_ASSIGN_TAB_2[2] = 3
```

```
MD 24300: TRAFO_TYP_3 = 514
MD 24310: TRAFO_AXES_IN_3[0] = 1
MD 24310: TRAFO_AXES_IN_3[1] = 4
MD 24310: TRAFO_AXES_IN_3[2] = 3
MD 24310: TRAFO_AXES_IN_3[3] = 0
MD 24310: TRAFO_AXES_IN_3[4] = 0
MD 24320: TRAFO_GEOAX_ASSIGN_TAB_3[0] = 1
MD 24320: TRAFO_GEOAX_ASSIGN_TAB_3[1] = 4
MD 24320: TRAFO_GEOAX_ASSIGN_TAB_3[2] = 3

                ; Data for TRACON
;TRACON chaining TRANSMIT 514 / TRAANG(Y1 axis to X1)

MD 24400: TRAFO_TYP_4 = 8192
MD 24995: TRACON_CHAIN_1[0] = 3
MD 24995: TRACON_CHAIN_1[1] = 1
MD 24995: TRACON_CHAIN_1[2] = 0
MD 24420: TRAFO_GEOAX_ASSIGN_TAB_4[0] = 1
MD 24420: TRAFO_GEOAX_ASSIGN_TAB_4[1] = 4
MD 24420: TRAFO_GEOAX_ASSIGN_TAB_4[2] = 3

                ; TRACON chaining TRANSMIT 257 /
                ; TRAANG(Y1 axis inclined in relation to X1)

MD 24430: TRAFO_TYP_5 = 8192
MD 24996: TRACON_CHAIN_2[0] = 2
MD 24996: TRACON_CHAIN_2[1] = 1
MD 24996: TRACON_CHAIN_2[2] = 0
MD 24434: TRAFO_GEOAX_ASSIGN_TAB_5[0] = 1
MD 24434: TRAFO_GEOAX_ASSIGN_TAB_5[1] = 4
MD 24434: TRAFO_GEOAX_ASSIGN_TAB_5[2] = 3
M17
```

; Appropriate **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              ; TRAANG(,1) not necessary because automatically
                   ; selected
N1020 X10 Y20 Z30
N1110 TRANSMIT(1)  ; TRACON(2) not necessary because automatically
                   ; converted
N1120 X10 Y20 Z30
N1130 Y2=0
N1210 TRAFOOF      ; TRAANG(,1) not necessary because automatically
                   ; converted
N1220 X10 Y20 Z30
M30
```

## 2.5 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**

The "handling transformation package" is required to implement cartesian PTP travel. Machine data 24100: 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 travel is active, axes in the transformation
which are being traversed, e.g. by the POS command, cannot be configured simultaneously as 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)

– Command PTPG0:
  The programmed cartesian PTP motion is executed automatically in every 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. Via the MD 20152: GCODE_RESET_VALUES[48] can be used to switch the default setting to cartesian PTP travel.

**Reset**

MD 20152: GCODE_RESET_MODE[48] (group 49) defines which setting is active after RESET/end of part program.

– MD=0: The setting depends on machine data
  MD 20150:GCODE_RESET_VALUES[48]

– MD=1: Active setting remains valid

**Selection**

With MD 20152: GCODE_RESET_MODE[48] set to 0,
the following can be activated with MD 20150: GCODE_RESET_VALUES[48]

– 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 subroutines 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 in combination with PTP. G0 and G41 are not basically mutually exclusive. However, an active PTP generates different contours to those computed for the TRC, resulting in the activation of a TRC alarm.
- When PTPG0 and a tool radius compensation are active, the CP command is applied.
Since G0 and G41 are not mutually exclusive, the CP command is automatically selected when a tool radius compensation is active. The radius compensation therefore works on the basis of clearly defined contours.
- Smooth approach and retraction (SAR) cannot be combined with PTP. SAR requires a contour to be able to construct the approach and retract motions. This information is not available with PTP.
- When smooth approach and retraction (SAR) and PTPG0 are active, the CP travel command is applied.
SAR requires a contour to be able to construct the approach and retract motions, and to position and lift off 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.
- Stock removal cycles such as CONTPRON, CONTDCON cannot be combined with PTP.
Stock removal cycles require a contour to be able 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 stock removal cycles such as CONTPRON, 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 points to be noted**

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 with a block search can produce different machine axis positions for the same cartesian position if a program section is executed with block search.

**Interrupts**

An illegal action which might cause a conflict is rejected with the following alarms:

| | |
|---|---|
| Alarm 14144: | If a TRC is selected or activated in PTP.<br>Also when PTP is combined with smooth approach and retraction (SAR) or PTP without the necessary G0 and G1 blocks. |
| Alarm 10753: | When PTPG0 and TRC are active, CP is selected internally to ensure that the tool radius compensation is executed correctly. |
| Alarm 10754: | Possible if conflict does occur. |
| Alarm 10778: | Possible if conflict does occur. |
| Alarm 10744: | When PTPG0 and SAR are active, CP command is used so that smooth approach and retraction is correctly executed. |
| Alarm 10746: | Possible if conflict does occur. |
| 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:** /PGA/, Programming Guide Advanced,
Section Transformations, “Cartesian PTP Travel”

---

## 2.5.1 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 a unique axis angle. 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 entitled "Handling Transformation Package (TE4)", the bits are assigned to different joint positions, as shown in Fig. 2-23. See also Subsection 2.5.3.

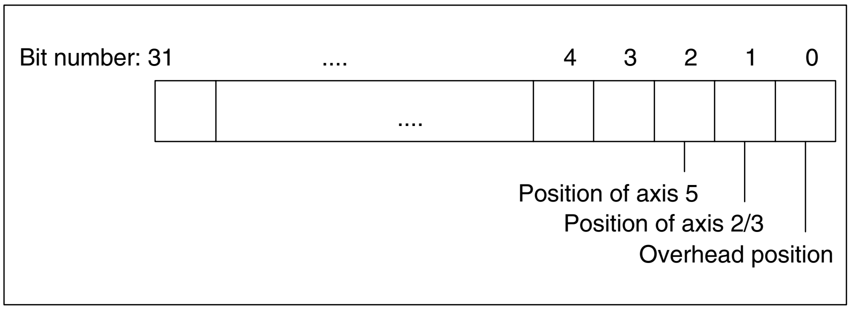



Fig. 2-23 Position bits for Handling Transformation Package

---

**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.

---

### 2.5.2 Overlap areas of axis angles

**TU address**

In order to approach axis angles in excess of ± 180 degrees without ambiguity, the information must be programmed in the TU (turn) address. The TU address thus represents the sign of the axis angles. An axis angle of $|\theta| < 360$ degrees can therefore be approached without ambiguity.

Variable TU contains a bit, which indicates the traversing direction for every axis involved in the transformation.

– TU bit=0: 0 degrees $\leq \theta <$ 360 degrees

– TU bit=1: –360 degrees $< \theta <$ 0 degrees

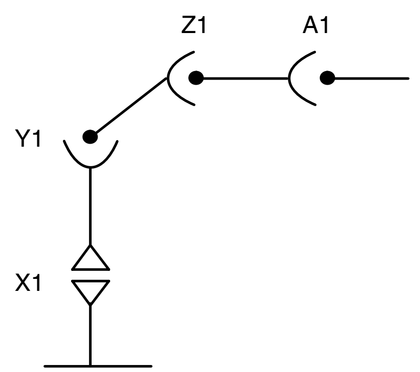
The TU bit is set to 0 for linear axes.

In the case of axes with a traversing range > ± 360 degrees, 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.

### 2.5.3 Examples of ambiguities of position

The kinematics for a 6axis joint have been used to illustrate the ambiguities caused by different joint positions.




Position 1:
X1=0°
Y1=45°
Z1=45°
STAT='B00'

Position 2:
X1=–180°
Y1=–45°
Z1=–45°
STAT='B01'

Fig. 2-24 Ambiguity in overhead area

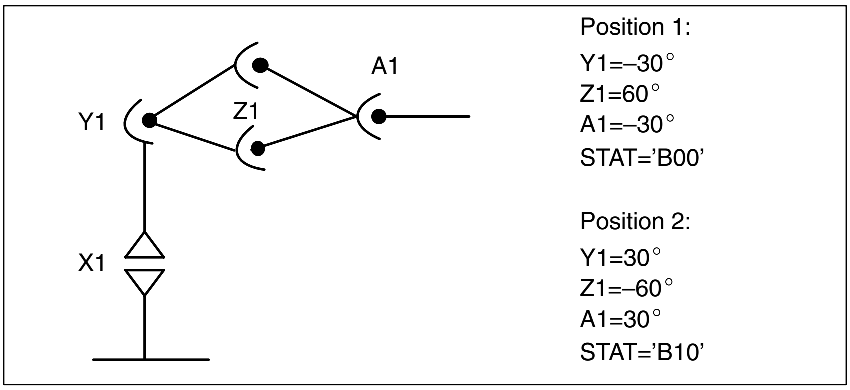



Fig. 2-25 Ambiguity of top or bottom elbow

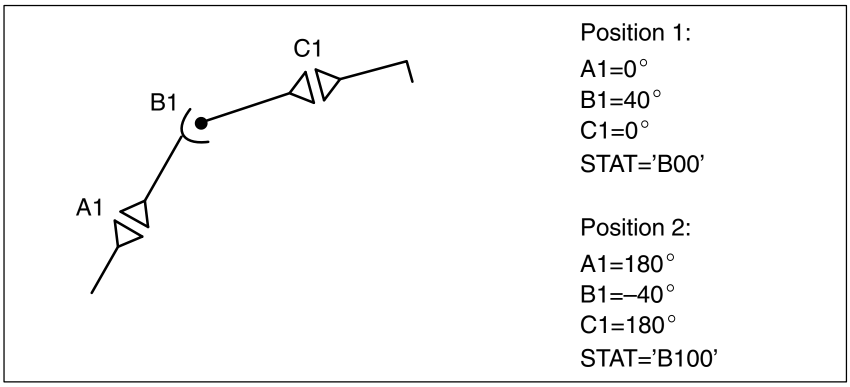



Fig. 2-26 Ambiguity of axis B1

## 2.5.4 Example of ambiguity in rotary axis position

The rotary axis position shown in Fig. 2-27 can be approached in negative or positive direction. The direction is programmed under address A1.

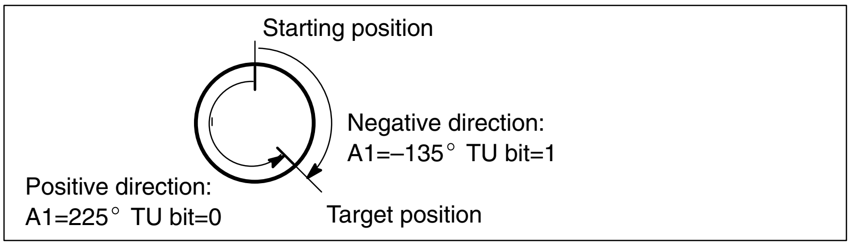



Fig. 2-27 Ambiguity in rotary axis position

## 2.5.5 PTP/CP switchover in JOG mode

In JOG mode, the transformation can be switched on and off via a PLC control signal. This control signal is active only in JOG mode and when a transformation has been activated via the program.

If the mode is switched back to AUTO, the state which was last active before switchover is made active again.

The “point-to-point traversal active” signal DBX317.6 shows which traversal type is active. By means of the “Activate point-to-point traversal” signal DBX29.4 the traversal type can be modified.

**Mode change**

The “Cartesian PTP travel” function can be used meaningfully only in the AUTO and MDA modes. The CP setting is automatically activated if the operating mode is switched to JOG. If the mode is then switched back to AUTO or MDA, 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 takes place in PTP. For a sloping axis “TRAANG”, only CP travel is active in REPOS mode.

## 2.6 Cartesian manual travel (810D powerline, SW 6.1 and higher)

### Functionality

The Cartesian manual travel function allows you to set axes independently in the Cartesian coordinate systems in order to provide a reference system for JOG mode.

- Basic coordinate system BCS MD 21106: Bit0 = 1
- Workpiece coordinate system WCS MD 21106: Bit1 = 1
- Tool coordinate system TCS MD 21106: Bit2 = 1

Machine data MD 21106: CART_JOG_SYSTEM, which is also used to activate the Cartesian manual travel function, is used for this purpose.

---

**Note**

The Cartesian manual travel function is implemented in SINUMERIK 810D powerline with CCU3 SW 6.1 and higher. SINUMERIK 840D requires the "handling transformation package" option SW6.3 or higher.

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

**References:** /FB1/, Description of Functions, Basic Machine, K2 Axes, Coordinate Systems, Frames, Reset Behavior

---

Representation of the reference system in the coordinate system:

| | | | |
|---|---|---|---|
|  | **WCS** = Workpiece zero |  | **TCS** = Tool reference point |

### Selecting reference systems

For JOG motion, you can specify one of three reference systems separately both for

**Translation** (coarse shift) of the geometry axes and for
**Orientation** for orientation axes using

SD 42650: CART_JOG_MODE.

If more than one bit is set for the translation or for the orientation reference system, or if an attempt is made to set a reference system that has not been activated via MD 21106: CART_JOG_SYSTEM, alarm 14148 "Reference system not permissible for Cartesian manual travel" is issued.

### 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.

The machine data MD 24120: 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.

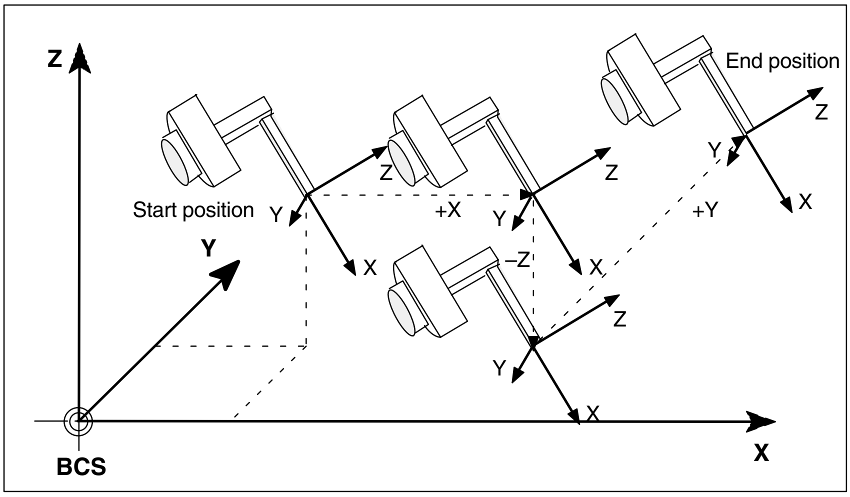



Fig. 2-28 Cartesian manual travel in the basic coordinate system (translation)

**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.

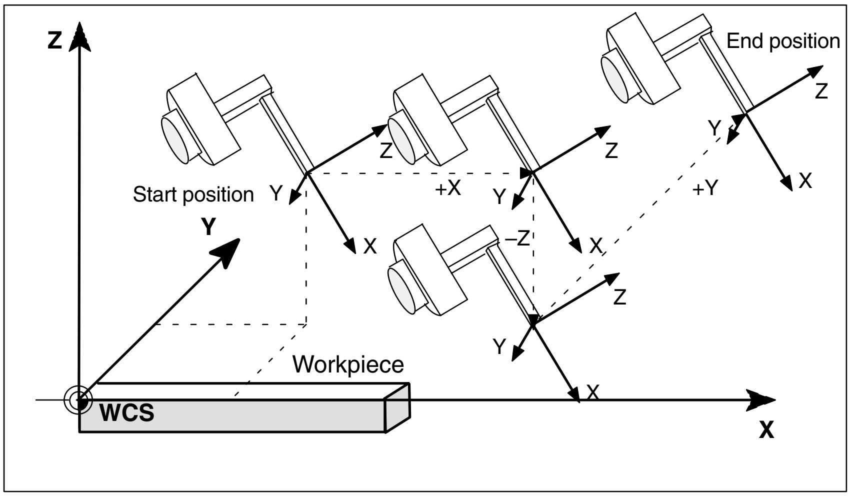



Fig. 2-29 Cartesian manual travel in the workpiece coordinate system (translation)

**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.

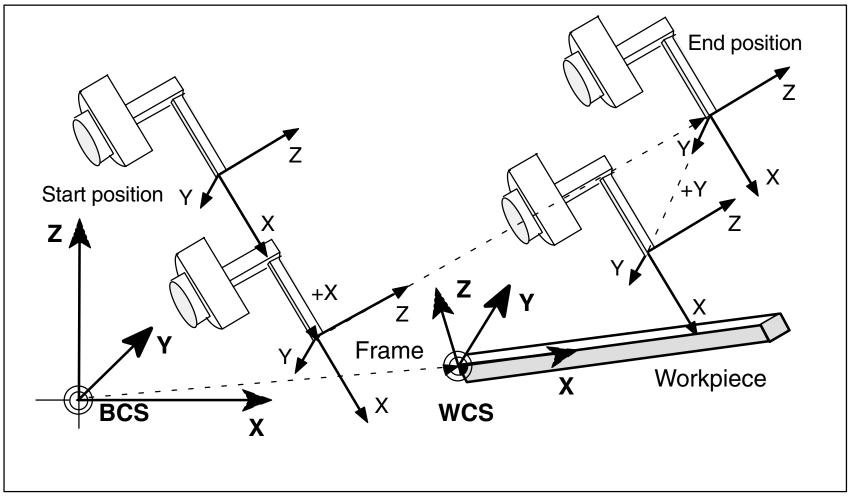



Fig. 2-30 Cartesian manual travel in the tool coordinate system (translation)

**Translation and orientation in the TCS simultaneously**

If translation and orientation motions 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 through the Z axis
- B angle : Rotation through the Y axis
- C angle : Rotation through 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

ORIEULER, ORIRPY, ORIVIRT1 and ORIVIRT2.

With ORIVIRT1, rotations are executed according to MD 21120: ORIAX_TURN_TAB_1. The orientation axes are assigned to the channel axes via MD 24585: 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.

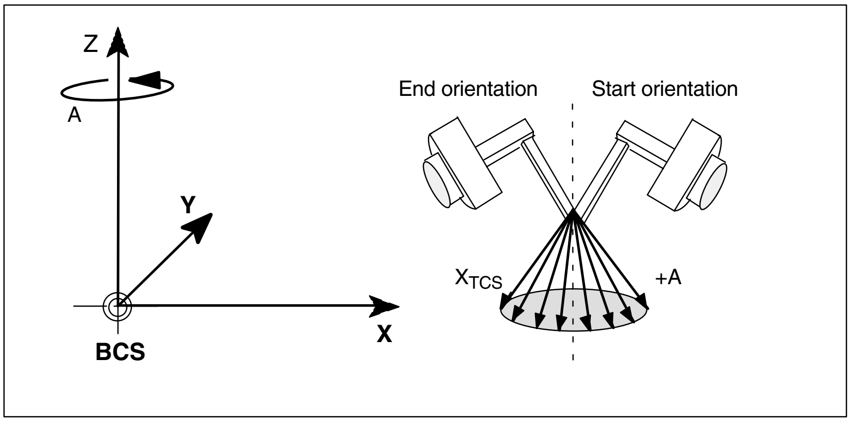



Fig. 2-31 Cartesian manual travel in the basic coordinate system orientation angle A

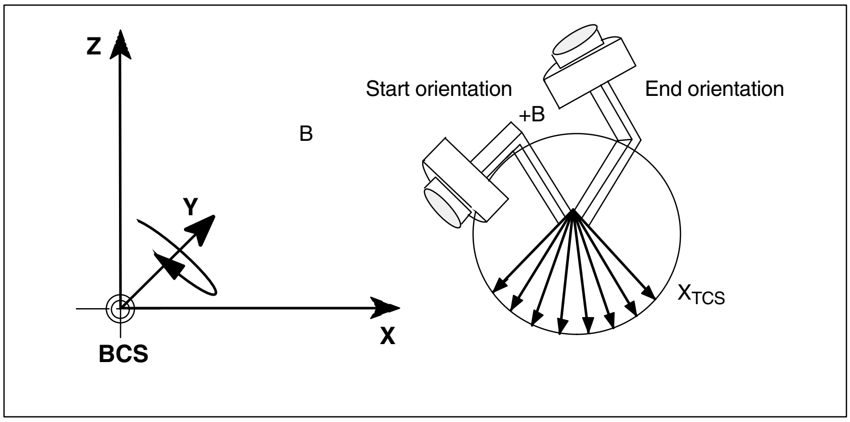



Fig. 2-32 Cartesian manual travel in the basic coordinate system orientation angle B

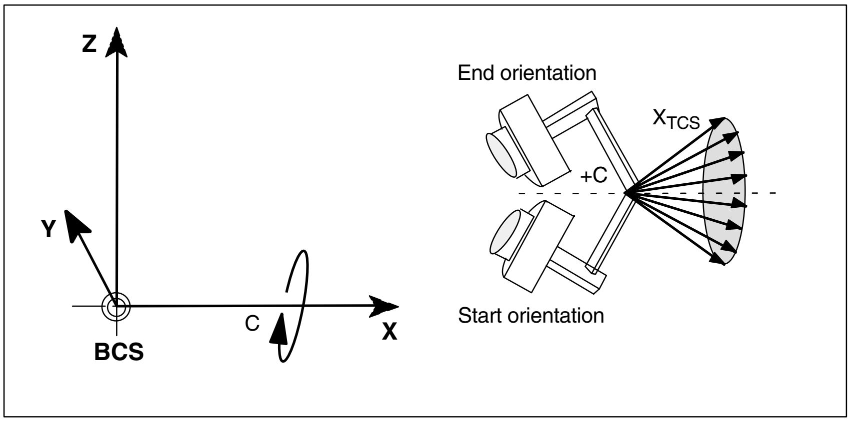



Fig. 2-33 Cartesian manual travel in the basic coordinate system orientation angle C

**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.

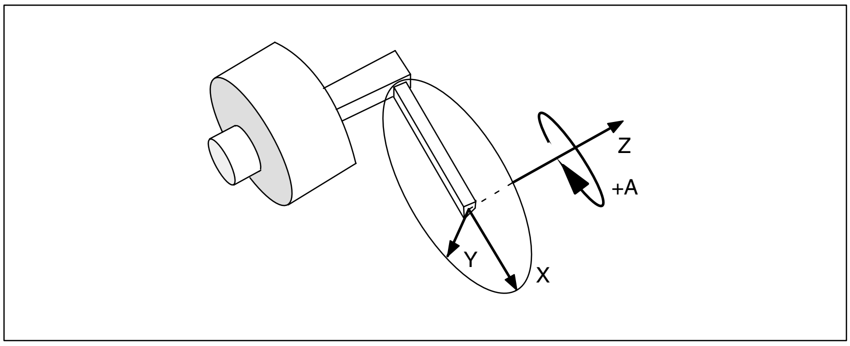



Fig. 2-34 Cart. manual travel in the tool coordinate system, orientation angle A

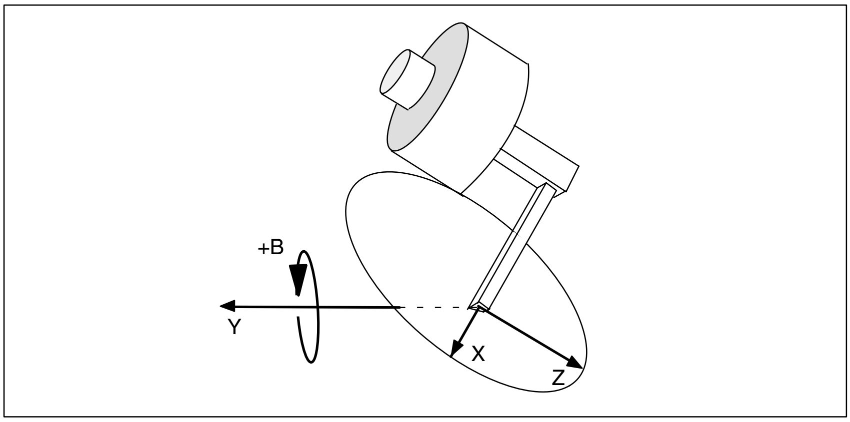



Fig. 2-35 Cart. manual travel in the tool coordinate system, orientation angle B

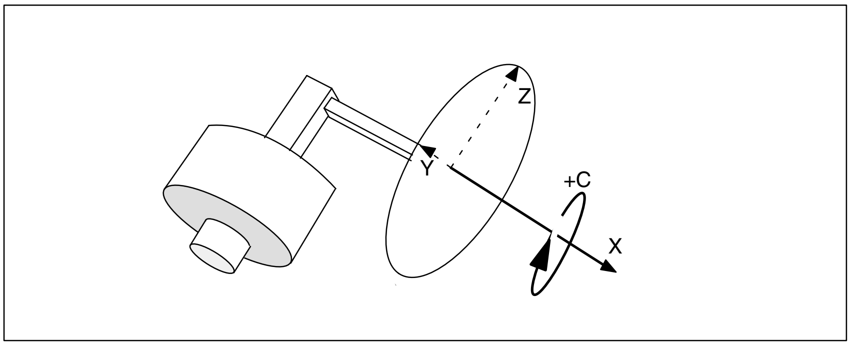



Fig. 2-36 Cart. manual travel in the tool coordinate system, orientation angle C

### Supplementary conditions

The cartesian manual travel function can be executed only when IS “Transformation active” (DB31, ... DBX33.6) is set to 1. The following supplementary conditions apply:

- SINUMERIK 840D requires the “handling transformation package” option with 5-axis or 6-axis transformation SW6.3 and higher.
- Virtual orientation axes must be defined via machine data MD 24585: TRAFO5_ORIAX_ASSIGN_TAB_1[n].
- The IS “Activate PTP/CP travel” (DB31, ... DBX29.4) must be 0.
- Machine data MD 21106: CART_JOG_SYSTEM must be > 0.

Table 2-3 Conditions for Cartesian manual travel

| Transformation in program active (TRAORI..) | G codes PTP/CP | IS “Activate PTP/CP traversing” | IS ‘Transformation active” |
|---|---|---|---|
| FALSE | Not functional! | Not functional! | DB31, ... DBX33.6 = 0 |
| TRUE | CP | DB31, ... DBX29.4 = 0 | DB31, ... DBX33.6 = 1 |
| TRUE | CP | DB31, ... DBX29.4 = 1 | DB31, ... DBX33.6 = 0 |
| TRUE | PTP | DB31, ... DBX29.4 = 0 | DB31, ... DBX33.6 = 1 |
| TRUE | PTP | DB31, ... DBX29.4 = 1 | DB31, ... DBX33.6 = 0 |

The G code PTP/CP currently active in the program does not affect Cartesian manual travel. The VDI 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 machine data MD 21106: CART_JOG_SYSTEM > 0.

  The BCS, WCS or TCS reference systems are enabled by setting the bits in MD 21106: CART_JOG_SYSTEM.
- The JOG traversing motion via SD 42650: CART_JOG_MODE

  Standard response 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.

The meaning of the bits is explained in the table below 2-4.

Table 2-4 Bit assignment for SD 42650: CART_JOG_MODE (only one bit may be set)

| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | Translation in the TCS | Translation in the WCS | Translation in the BCS |
| **Bit15** | **Bit14** | **Bit13** | **Bit12** | **Bit11** | **Bit10** | **Bit9** | **Bit8** |
| Reserved | | | | | Orientation in TCS | Orientation in WCS | Orientation in BCS |

**Combining reference systems**

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

Table 2-5 Combination options for reference systems

| SD 42650: CART_JOG_MODE | | | | | | Reference system for | |
|---|---|---|---|---|---|---|---|
| Bit 10 | Bit 9 | Bit 8 | Bit 2 | Bit 1 | Bit 0 | Orientation | Translation |
| 0 | 0 | 0 | don't care | don't care | don't care | 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 |

## 2.7 Activating the transformation MD via parts program/softkey (SW 5.2 and later)

### 2.7.1 Functionality

**SW 5.1 and lower**

Up to eight different transformations can be set in the control in SW 5.1 and lower. The transformation type is set in machine data $MC_TRAFO_TYPE_1 to $MC_TRAFO_TYPE_8.

For each transformation group (TRANSMIT, see Section 2.1), TRACYL (see Section 2.2), TRAANG (see Section 2.3) and chained transformations (see Section 2.4) there are two transformation data sets, i.e. no more than two transformations can be set from one group, even when the eight available transformations have not yet all been programmed.

**As of SW 5.2**

Transformation MD can now be activated by means of a program command softkey (NEWCONFIG-capable), i.e. these can, for example, be written from the parts program, thus altering the transformation configuration completely. The specified restrictions regarding the number of available transformations thus no longer apply.

---

**Note**

However, the number of transformation machine data sets is limited as in previous versions.

---

**As of SW 7.2**

Up to ten different transformations can be set in the control in SW 7.2 and higher. The transformation type is set in machine data $MC_TRAFO_TYPE_1 to $MC_TRAFO_TYPE_10.

**Features**

The machine data listed in Section 4.3 were activated by power ON in SW 5.1 and lower. They are NEWCONFIG-capable in SW 5.2 and higher.
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 NEWCONFIG command is issued (regardless whether via the NEWCONF NC program command, the MMC 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.

### 2.7.2 Supplementary conditions

**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). It would be permissible, for example, with an active transformation parameterized in MD 24100: TRAFO_TYPE = 16 (5-axis transformation with rotatable tool and two mutually perpendicular rotary axes A and B) to change machine data $MC_TRAFO5_NUTATOR_AX_ANGLE_n since this particular machine data is not involved in the transformation.

Please also note that machine data MD 21110: X_AXIS_IN_OLD_X_Z_PLANE must not be changed when an orientation transformation is active.

---

**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. MD 24100: TRAFO_TYPE_1 = 16, MD 24200: 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. MD 24500: TRAFO5_PART_OFFSET_1, but not e.g. MD 24650: TRAFO5_BASE_TOOL_2 or
MD 21110: X_AXIS_IN_OLD_X_Z_PLANE

Furthermore, MD 24300: TRAFO_TYPE_3 = 256, for example, can be used to set another transformation (Transmit) which is parameterized by other machine data.

**Defining geometry axes**

Geometry axes must be defined in $MC_TRAFO_GEOAX_ASSIGN_TAB_X[n] or MD 20050: AXCONF_GEOAX_ASSIGN_TAB[n] **before** the control system powers up.

**Changing the assignment**

The assignment between a transformation data set and a transformation is determined by the sequence of entries in $MC_TRAFO_TYPE_X. The first transformation data set is assigned to the first entry in the table, the second data set to the second entry. 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.

MD 24100: TRAFO_TYPE_1 = 16 ; Orientation transformation,
; 1st orientation transformer data block

MD 24200: TRAFO_TYPE_2 = 256 ; Transmit transformation

```
MD 24300:TRAFO_TYPE_3 = 18   ; Orientation transformation,
                             ; 2nd orientation transformer data block
```

The first data set for orientation transformations is assigned to the first transformation (equalling the first orientation transformation) and the second transformation data set to the third transformation (equalling 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 this example, however, it is legal to set another orientation transformation as the first transformation (e.g. using MD 24100: TRAFO_TYPE_1 = 32) or to set a transformation from another group as the first transformation (e.g. using MD 24100: TRAFO_TYPE_1 = 1024, TRAANG) if the second transformation is changed to an orientation transformation at the same time, e.g. using MD 24200: TRAFO_TYPE_2 = 48.

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

Machine data MD 20110: RESET_MODE_MASK,
MD 20112: START_MODE_MASK and MD 20140: TRAFO_RESET_VALUE can be programmed to select a transformation automatically in response to RESET (i.e. at end of program as well) and / or on program start.

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:

```
N10   TRAFOOF()                         ; Deselect any active
                                        ; transformation
N20   $MC_TRAFO5_BASE_TOOL_1[0]=0       ; Write machine data
N30   $MC_TRAFO5_BASE_TOOL_1[0]=3       ;
N40   $MC_TRAFO5_BASE_TOOL_1[0]=200;


N130  NEWCONF                           ; Accept newly modified
                                        ; machine data
N140  M30
```

### 2.7.4 List of machine data affected

The machine data which can be made NEWCONFIG-capable are listed below.

**All transformations**

Machine data which are relevant for all transformations:

- $MC_TRAFO_TYPE_1 to $MC_TRAFO_TYPE_10
- $MC_TRAFO_AXES_IN_1 to $MC_TRAFO_AXES_IN_10
- $MC_TRAFO_GEOAX_ASSIGN_TAB_1 to
  $MC_TRAFO_GEOAX_ASSIGN_TAB_10

**Orientation transformations**

Machine data which are relevant for orientation transformations:

- MD 24550: TRAFO5_BASE_TOOL_1 and
  MD 24650: TRAFO5_BASE_TOOL_2
- MD 24558: TRAFO5_JOINT_OFFSET_1 and
  MD 24658: TRAFO5_JOINT_OFFSET_2
- MD 24500: TRAFO5_PART_OFFSET_1 and
  MD 24600: TRAFO5_PART_OFFSET_2
- MD 24510: TRAFO5_ROT_AX_OFFSET_1 and
  MD 24610: TRAFO5_ROT_AX_OFFSET_2
- MD 24520: TRAFO5_ROT_SIGN_IS_PLUS_1 and
  MD 24620: TRAFO5_ROT_SIGN_IS_PLUS_2
- MD 24530: TRAFO5_NON_POLE_LIMIT_1 and
  MD 24630: TRAFO5_NON_POLE_LIMIT_2
- MD 24540: TRAFO5_POLE_LIMIT_1 and
  MD 24640: TRAFO5_POLE_LIMIT_2
- MD 24570: TRAFO5_AXIS1_1 and MD 24670: TRAFO5_AXIS1_2
- MD 24572: TRAFO5_AXIS2_1 and MD 24672: TRAFO5_AXIS2_2
- MD 24574: TRAFO5_BASE_ORIENT_1 and
  MD 24674: TRAFO5_BASE_ORIENT_2
- MD 24562: TRAFO5_TOOL_ROT_AX_OFFSET_1 and
  MD 24662: TRAFO5_TOOL_ROT_AX_OFFSET_2
- MD 24564: TRAFO5_NUTATOR_AX_ANGLE_1 and
  MD 24 664: TRAFO5_NUTATOR_AX_ANGLE_2
- MD 24566: TRAFO5_NUTATOR_VIRT_ORIAX_1 and
  MD 24666: TRAFO5_NUTATOR_VIRT_ORIAX_2

**Transmit transformations**

Machine data which are relevant for Transmit transformations:

- MD 24920: TRANSMIT_BASE_TOOL_1 and
  MD 24970: TRANSMIT_BASE_TOOL_2
- MD 24900: TRANSMIT_ROT_AX_OFFSET_1 and
  MD 24950: TRANSMIT_ROT_AX_OFFSET_2
- MD 24910: TRANSMIT_ROT_SIGN_IS_PLUS_1 and
  MD 24960: TRANSMIT_ROT_SIGN_IS_PLUS_2
- MD 24911: TRANSMIT_POLE_SIDE_FIX_1 and
  MD 24961: TRANSMIT_POLE_SIDE_FIX_2

**Tracyl transformations**

Machine data which are relevant for Tracyl transformations:

- MD 24820: TRACYL_BASE_TOOL_1 and
  MD 24870: TRACYL_BASE_TOOL_2
- MD 24800: TRACYL_ROT_AX_OFFSET_1 and
  MD 24850: TRACYL_ROT_AX_OFFSET_2
- MD 24810: TRACYL_ROT_SIGN_IS_PLUS_1 and
  MD 24870: TRACYL_ROT_SIGN_IS_PLUS_2
- MD 24808: TRACYL_DEFAULT_MODE_1 and
  MD 24858: TRACYL_DEFAULT_MODE_2

**Inclined axis transformations**

Machine data which are relevant for inclined axes:

- MD 24710: TRAANG_BASE_TOOL_1 and
  MD 24760: TRAANG_BASE_TOOL_2
- MD 24700: TRAANG_ANGLE_1 and MD 24750: TRAANG_ANGLE_2
- MD 24720: TRAANG_PARALLEL_VELO_RES_1 and
  MD 24770: TRAANG_PARALLEL_VELO_RES_2
- MD 24721: TRAANG_PARALLEL_ACCEL_RES_1 and
  MD 24771: TRAANG_PARALLEL_ACCEL_RES_2

**Chained transformations**

Machine data which are relevant for chained transformations:

- MD 24995: TRACON_CHAIN_1 and MD 24996: TRACON_CHAIN_2
- MD 24997: TRACON_CHAIN_3 and MD 24998: TRACON_CHAIN_4

**Persistent transformation**

Machine data which are relevant for persistent transformations:

- MD 20144: TRAFO_MODE_MASK
- MD 20140: TRAFO_RESET_VALUE
- MD 20110: RESET_MODE_MASK and MD 20112: START_MODE_MASK

**Not transformation specific**

Machine data that are not transformation-specific, i.e. they are not uniquely assigned to a particular transformation data set or they are relevant even when a transformation is not active:

- MD 21110: X_AXIS_IN_OLD_X_Z_PLANE
- MD 21090: MAX_LEAD_ANGLE
- MD 21092: MAX_TILT_ANGLE
- MD 21100: ORIENTATION_IS_EULER

■

# Notes

# Supplementary Conditions 3

## 3.1 TRANSMIT

**Availability**

The “TRANSMIT” function is an option with order number:
6FC5 251-0AB01-0AA0

It is available from product version 2 onwards for:

- SINUMERIK 840D with NCU 571–573
- SINUMERIK 810D

Pole traversal and optimized control response in pole vicinity are available with SW 4.1 and higher.

## 3.2 TRACYL (peripheral surface transformation)

**Availability**

The “TRACYL” function is an option with order number:
6FC5 251-0AB01-0AA0

It is available from software version 2 onwards for:

- SINUMERIK 840D with NCU 571–573
- SINUMERIK 810D

## 3.3 TRAANG (inclined axis)

**Availability**

The “TRAANG” function (inclined axis) is an option with order number:
6FC5 251-0AB06-0AA0

It is available from software version 2 onwards for:

- SINUMERIK 840D with NCU 572–573.2
- SINUMERIK 810D

# 3.4 Chained transformations (SW 5)

**SW 5 and higher**

**Two** transformations can be chained.

However, not just any transformation can be chained to another one.
The following restrictions apply in SW version 5:
- The **first** transformation in the chain must be:
  - an orientation transformation
    (3, 4, 5-axis transformation, universal milling head)
  - Transmit or
  - Peripheral transformation or
  - Inclined axis
- The **second** transformation must be an **inclined** axis transformation.
- Only two transformations may be chained.

It is permissible (e.g. for test purposes) to enter only one transformation in the chain list.

■

# Data Descriptions (MD, SD) 4

## 4.1 Channelspecific machine data

<table>
<tr><td colspan="2">21110<br>MD number</td><td colspan="4">X_AXIS_IN_OLD_X_Z_PLANE<br>Coordinate system with automatic FRAME definition</td></tr>
<tr><td colspan="2">Default setting: 1</td><td colspan="2">Minimum input limit: 0</td><td colspan="2">Maximum input limit: 1</td></tr>
<tr><td colspan="2">Changes effective after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (up to SW 5.1)</td><td colspan="3">Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1)</td><td>Unit: –</td></tr>
<tr><td colspan="3">Data type: BOOLEAN</td><td colspan="3">Applies from SW: 2</td></tr>
<tr><td>Meaning:</td><td colspan="5">1 = With automatic definition of a frame (TOFRAME) whose Z direction is the same as the actual tool orientation, the new coordinate system is rotated additionally around the new Z axis with the result that the new X axis lies in the old Z/X plane.<br>0 = With automatic definition of a frame (TOFRAME) whose Z direction is the same as the actual tool orientation, the new coordinate system is left as it is represented by the machine kinematics, i.e. the coordinate system rotates with the tool (orientation).</td></tr>
<tr><td>MD irrelevant for ......</td><td colspan="5">No orientation programming.</td></tr>
<tr><td>Related to ....</td><td colspan="5">MD 21100</td></tr>
<tr><td>References</td><td colspan="5">Programming Guide</td></tr>
</table>

## 4.2 Transformation-specific machine data

| | | | | | |
|---|---|---|---|---|---|
| **20144** MD number | **TRAFO_MODE_MASK** Selection of the kinematic transformation function | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after RESET | | | Protection level: 2 / 7 | | Unit: – |
| Data type: Byte | | | | Applies from SW: 7.2 | |
| Meaning: | Selects a particular scope of kinematic transformation functions by setting the following bits:<br>Bit Hex. Meaning<br>0:(LB) 0x01 The transformation selected in MD 20140: TRAFO_RESET_VALUE is persistent, i.e. it is selected with TRAFOOF and is not shown in the display.<br><br>Meaning of individual bits:<br>Bit 1=0: Default behavior as in previous version<br><br>Bit 1=1: The transformation specified in MD 20140: TRAFO_RESET_VALUE is persistent, i.e. it is selected with TRAFOOF and is not shown in the display.<br>**Precondition for this option** is that the transformation selected in MD 20140: TROFO_RESET_VALUE is automatically selected via<br>MD 20110: RESET_MODE_MASK and<br>MD 20112: START_MODE_MASK on RESET and START,<br>i.e.:<br>MD 20110: RESET_MODE_MASK Bit 0=1 and Bit 7=0,<br>MD 20112: START_MODE_MASK Bit 7=1<br>MD 20118: GEOAX_CHANGE_REST=TRUE | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **22534** MD number | **TRAFO_CHANGE_M_CODE** M code for transformation changeover | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 99999999 | |
| Change applies from: | | | Protection level: 2 / 7 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 4.1 | |
| Meaning: | Number of M code that is output at the VDI interface in the case of a transformation changeover on the geometry axes.<br>No M code is output if this MD is set to one of the values 0 to 6, 17 or 30.<br>It is not monitored whether an M code created in this way will conflict with other functions. | | | | |

| | | | |
|---|---|---|---|
| **24100**<br>MD number | **TRAFO_TYPE_1**<br>Type of 1st transformation | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: – | |
| Changes eff. after NEWCONFIG (SW 5.2 and higher)<br>Power ON (SW 5.1 and lower) | Prot. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: – | |
| Data type: DWORD | Applies from SW: 2.0 | | |
| Meaning: | This MD specifies for each channel which transf. is available as the first in the channel.<br>Identifier for specifying axis sequence in the case of 5-axis transformation and transformation type for each of the permissible transformations<br>Bits 4 … 3 … 0: Transformation type \| Axis sequence<br>**Transformation type:**<br>0 No transformation<br>16 5-axis transformation with rotatable tool<br>32 5-axis transformation with rotatable workpiece<br>48 5-axis transformation with rotatable tool and workpiece<br>**Axis sequence** for transformation types **16 –48**<br>0 Axis sequence AB<br>1 Axis sequence AC<br>2 Axis sequence BA<br>3 Axis sequence BC<br>4 Axis sequence CA<br>5 Axis sequence CB<br>8 Generic 5-axis transformation<br>256 TRANSMIT transformation<br>257 TRANSMIT transformation with additional linear axis<br>512 TRACYL transformation (generated cylinder surface transformation)<br>513 TRACYL transformation with X-Y-Z-C kinematics<br>514 TRACYL transformat. without groove side offset, with additional linear axis<br>1024 TRAANG transformation (inclined axis)<br>2048 Centerless transformation<br>8192 Chained transformation<br>When values are assigned to transformation types 16-48, the associated axis sequences are added. Axis sequences for transformation types 256 – 2048 are meaningless (no error message). | | |
| MD irrelevant for ... ... | No transformations | | |
| Application example(s) | $MC_TRAFO_TYP_1=20 ; (16+4) | | |
| Related to .... | TRAFO_TYPE_2, TRAFO_TYPE_3, ... TRAFO_TYPE_10 | | |
| References | /FB/, F2, “5-axis transformation” | | |

| | | | |
|---|---|---|---|
| **24110**<br>MD number | **TRAFO_AXES_IN_1[i]**<br>Axis assignment for transformation 1 [axis index]: 0 ... [max. number of channel axes] | | |
| Default setting: 1,2,3,4,5,0,0,0 | Minimum input limit: 0 | Maximum input limit: 10 [max. number of channel axes] | |
| Changes eff. after NEWCONFIG (SW 5.2 and higher)<br>Power ON (SW 5.1 and lower) | Prot. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: | |
| Data type: Byte | Applies from SW: 2.0 | | |
| Meaning: | Axis assignment at input of 1st transformation<br>Index i assumes the values 0, 1 and 2 with TRANSMIT.<br>The assignment for TRANSMIT is:<br>$MC_TRAFO_AXES_IN_1[0]= Channel axis number of axis perpendicular to rotary axis<br>$MC_TRAFO_AXES_IN_1[1]= Channel axis number of rotary axis<br>$MC_TRAFO_AXES_IN_1[2]= Channel axis number of axis parallel to rotary axis<br>The index entered at the nth position specifies which axis is mapped internally by the transformation on axis n. | | |
| MD irrelevant for ...... | No transformation | | |
| Application example(s) | $MC_TRAFO_AXES_IN_1[0]=1 | | |
| Related to .... | TRAFO_AXES_IN_2, TRAFO_AXES_IN_3, ... TRAFO_AXES_IN_10 | | |
| References | /FB/, F2, “5-axis transformation” | | |

| | | | | |
|---|---|---|---|---|
| **24120**<br>MD number | **TRAFO_GEOAX_ASSIGN_TAB_1[i]**<br>Assignment of geometry axes to channel axes with transformation 1 [geometry axis number]: 0 ... 2. | | | |
| Default setting: 0,0,0 | Minimum input limit: 0 | | Maximum input limit: [max. no. of channel axes] | |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>Power ON (SW 5.1 and lower) | | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: |
| Data type: Byte | | | Applies from SW: 2.0 | |
| Meaning: | This MD specifies the channel axes on which the axes of the Cartesian coordinate system<br>are mapped when transformation 1 is active.<br>Index i assumes the values 0, 1, 2 with TRANSMIT. It refers to the first, second and third geometry axis. | | | |
| MD irrelevant for ...... | No transformation | | | |
| Application example(s) | $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=channel axis number | | | |
| Related to .... | $MC_AXCONF_GEOAX_ASSIGN_TAB, if no transformation is active. | | | |
| References | /FB/, K2, “Coordinate Systems, Axis Types, Axis Configurations, Workpiece-Related Actual-Value System, External Zero Offset” | | | |

| | | | | |
|---|---|---|---|---|
| **24130**<br>MD number | **TRAFO_INCLUDES_TOOL_1[i]**<br>Tool handling with active transformation 1 [geometry axis number]: 0 ... 2. | | | |
| Default setting: TRUE | Minimum input limit: 0 | | Maximum input limit: [max. no. of channel axes] | |
| Changes effective after NEWCONFIG | | Protection level: 7 / 7 | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 5.2 | |
| Meaning: | MD specifies which tool is to be handled when transformation 1 is active. | | | |

| | | | | |
|---|---|---|---|---|
| **24200/24300/ 24400/24430/24440/ 24450/24460/ 24470/24480**<br>MD number | **TRAFO_TYPE_2 / _3 / _4 / _5/ _6/ _7/ _8/ _9/ _10** expanded to 10 with SW 7.2 and later<br><br>Type of transformation | | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: – | |
| Changes eff. after NEWCONFIG (SW 5.2 and higher)<br>Power ON (SW 5.1 and lower) | | Prot. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: – |
| Data type: DWORD | | Applies from SW: 2.0 | | |
| Meaning: | As for TRAFO_TYPE_1, but applies to transformation that is available as the second ... tenth transformation in the channel. | | | |

| | | | | |
|---|---|---|---|---|
| **24210/24310/ 24410/24432/24442/ 24452/24462/ 24472/24482**<br>MD number | **TRAFO_AXES_IN_2[i] / _3[i] / _4[i]/ _5[i] / _6[i] / _7[i] / _8[i] / _9[i] / _10[i]** extended to 10 with SW 7.2 and later<br><br>Axis assignment for transformation 2/3/4/5/6/7/8/9/10 [axis index]: 0 ... [max. no. of channel axes] | | | |
| Default setting: 1,2,3,4,5,0,0,0 | Minimum input limit: 0 | | Maximum input limit: [max. no. of channel axes] | |
| Changes eff. after NEWCONFIG (SW 5.2 and higher)<br>Power ON (SW 5.1 and lower) | | Prot. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: – |
| Data type: Byte | | Applies from SW: 2.0 | | |
| Meaning: | Axis assignment at input of 2nd to 10th transformation.<br>Same meaning as for TRAFO_AXES_IN_1. | | | |

| | | | | |
|---|---|---|---|---|
| **24220/24320/ 24420/24434/24444/ 24454/24464/ 24474/24484**<br>MD number | **TRAFO_GEOAX_ASSIGN_TAB_2[i] / _3[i] / _4[i]/_5[i] / _6[i] / _7[i]/ _8[i] / _9[i] / _10[i]** extended to 10 with SW 7.2 and later<br><br>Assignment of geometry axes to channel axes with transformation 2/3/4/5/6/7/8/9/10 [geometry axis number]: 0 ... 2. | | | |
| Default setting: 0,0,0 | Minimum input limit: 0 | | Maximum input limit: [max. no. of channel axes] | |
| Changes eff. after NEWCONFIG (SW 5.2 and higher)<br>Power ON (SW 5.1 and lower) | | Prot. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: – |
| Data type: Byte | | Applies from SW: 2.0 | | |
| Meaning: | The channel axes on which the axes of the Cartesian coordinate system are mapped when transformation 2 to 10 is active are set in this MD.<br>The meaning otherwise corresponds to TRAFO_GEOAX_ASSIGN_TAB_1. | | | |

| | | | | |
|---|---|---|---|---|
| **24230/24330/ 24426/24436/24446/ 24456/24466/ 24476/24486**<br>MD number | **TRAFO_INCLUDES_TOOL_2[i] / _3[i] / _4[i]/_5[i] / _6[i] / _7[i]/ _8[i] / _9[i] / _10[i]** extended to 10 with SW 7.2 and later<br><br>Tool handling with active transf. 2/3/4/5/6/7/8/9/10 [geometry axis number]: 0 ... 2. | | | |
| Default setting: TRUE | Minimum input limit: 0 | | Max. input limit: [max. no. of channel axes] | |
| Changes effective after NEWCONFIG | | Protect. level: 7 / 7 | | Unit: – |
| Data type: BOOLEAN | | Applies from SW: 5.2 | | |
| Meaning: | MD specifies which tool is to be handled when transformation 2 to 10 is active.<br>Its meaning otherwise corresponds to TRAFO_INCLUDES_TOOL_1. | | | |

# 4.3 Function-specific machine data

## 4.3.1 TRANSMIT

| | | | | |
|---|---|---|---|---|
| **24900**<br>MD number | **TRANSMIT_ROT_AX_OFFSET_1**<br>Position offset of rotary axis | | | |
| Default setting: 0 | | Minimum input limit: 0 | Maximum input limit: 360 | |
| Changes effective after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | | Protection level: 2/4 | | Unit: Degrees |
| Data type: DOUBLE | | | Applies from SW: 2.0 | |
| Meaning: | Specifies the offset of the rotary axis in degrees in relation to the zero position while TRANSMIT is active for the first declared TRANSMIT transformation for each channel. | | | |
| MD irrelevant for ...... | No TRANSMIT active | | | |
| Application example(s) | $MC_TRANSMIT_ROT_AX_OFFSET_1=15.0 | | | |
| Related to .... | TRANSMIT_ROT_AX_OFFSET_2 | | | |

| | | | | |
|---|---|---|---|---|
| **24910**<br>MD number | **TRANSMIT_ROT_SIGN_IS_PLUS_1**<br>Sign of rotary axis 1/2 | | | |
| Default setting: 1 | | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | | Protection level: 2/4 | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 2.0 | |
| Meaning: | Specifies the sign which is applied to the rotary axis during the TRANSMIT transformation for the first declared TRANSMIT transformation for each channel. | | | |
| MD irrelevant for ...... | No TRANSMIT active | | | |
| Application example(s) | $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1=TRUE | | | |
| Related to .... | TRANSMIT_ROT_SIGN_IS_PLUS_2 | | | |

| | | | | |
|---|---|---|---|---|
| **24911**<br>MD number | **TRANSMIT_POLE_SIDE_FIX_1**<br>Restriction of working range in front of/behind pole, 1st transformation | | | |
| Default setting: 0 | | Minimum input limit: 0 | Maximum input limit: 2 | |
| Changes effective after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | | Protection level: 2/4 | | Unit: |
| Data type: BYTE | | | Applies from SW: 4.1 | |
| Meaning: | Restriction of working range in front of/behind pole or no restrictions, i.e. traversal through pole.<br>The assigned values have the following meanings:<br>1: Working range of linear axis for positions >=0,<br>(if tool length compensation parallel to linear axis equals 0)<br>2: Working range of linear axis for positions <=0,<br>(if tool length compensation parallel to linear axis equals 0)<br>0: No restriction of working range. Traversal through pole. | | | |

| | | | |
|---|---|---|---|
| **24920**<br>MD number | **TRANSMIT_BASE_TOOL_1[i]**<br>Vector of base tool on activation of transformation | | |
| Changes effective after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit:<br>mm |
| Data type: DOUBLE | | Applies from SW: 2.0 | |
| Meaning: | MD specifies the distance of the tool zero point referred to the appropriate geometry axes valid with TRANSMIT active and without tool length offset selected for the 1st TRANSMIT transformation for each channel.<br>Programmed length compensations are added to the base tool.<br>Index i assumes values 0, 1, 2 for the 1st to 3rd geometry axes. | | |
| MD irrelevant for ...... | No TRANSMIT active | | |
| Application example(s) | $MC_TRANSMIT_BASE_TOOL_1[0]=20.0 | | |
| Related to .... | $MC_TRANSMIT_BASE_TOOL_2 | | |

| | | | |
|---|---|---|---|
| **24950**<br>MD number | **TRANSMIT_ROT_AX_OFFSET_2**<br>Position offset of rotary axis | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: 360 |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit:<br>Degrees |
| Data type: DOUBLE | | Applies from SW: 2.0 | |
| Meaning: | Specifies the offset of the rotary axis in degrees in relation to the zero position while TRANSMIT is active for the second declared TRANSMIT transformation for each channel. | | |
| MD irrelevant for ...... | No TRANSMIT active | | |
| Related to .... | TRANSMIT_ROT_AX_OFFSET_1 | | |

| | | | |
|---|---|---|---|
| **24960**<br>MD number | **TRANSMIT_ROT_SIGN_IS_PLUS_2**<br>Sign of rotary axis 1/2 | | |
| Default setting: 1 | Minimum input limit: 0 | | Maximum input limit: 1 |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect.level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: – |
| Data type: BOOLEAN | | Applies from SW: 2.0 | |
| Meaning: | Specifies the sign which is applied to the rotary axis during the TRANSMIT transformation for the second declared TRANSMIT transformation for each channel. | | |
| MD irrelevant for ...... | No TRANSMIT active | | |
| Application example(s) | $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1=TRUE | | |
| Related to .... | TRANSMIT_ROT_SIGN_IS_PLUS_1 | | |

| | | | |
|---|---|---|---|
| **24961**<br>MD number | **TRANSMIT_POLE_SIDE_FIX_2**<br>Restriction of working range in front of/behind pole, 2nd transformation | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: 2 |
| Changes effective after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: – |
| Data type: BYTE | | Applies from SW: 4.1 | |
| Meaning: | Restriction of working range in front of/behind pole or no restrictions, i.e. traversal through pole.<br>The assigned values have the following meanings:<br>1: Working range of linear axis for positions >=0,<br>(if tool length compensation parallel to linear axis equals 0)<br>2: Working range of linear axis for positions <=0,<br>(if tool length compensation parallel to linear axis equals 0)<br>0: No restriction of working range. Traversal through pole. | | |

| | | | | |
|---|---|---|---|---|
| **24970**<br>MD number | **TRANSMIT_BASE_TOOL_2[i]**<br>Vector of base tool on activation of transformation | | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: | |
| Changes effect.. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: mm |
| Data type: DOUBLE | | Applies from SW: 2.0 | | |
| Meaning: | MD specifies the distance of the tool zero point referred to the appropriate geometry axes valid with TRANSMIT active and without tool length offset selected for the 1st TRANSMIT transformation for each channel.<br>Programmed length compensations are added to the base tool.<br>Index i assumes values 0, 1, 2 for the 1st to 3rd geometry axes. | | | |
| MD irrelevant for ...... | No TRANSMIT active | | | |
| Application example(s) | $MC_TRANSMIT_BASE_TOOL_2[0]=tx | | | |
| Related to .... | $MC_TRANSMIT_BASE_TOOL_1 | | | |

## 4.3.2 TRACYL

| | | | | |
|---|---|---|---|---|
| **24800**<br>MD number | **TRACYL_ROT_AX_OFFSET_1**<br>Offset of rotary axis for the 1st TRACYL transformation | | | |
| Default setting: 0 | Minimum input limit: [no limit] | | Maximum input limit: [no limit] | |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | | Protect. level:7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: Degrees |
| Data type: DOUBLE | | Applies from SW: 2.0 | | |
| Meaning: | Specifies the offset of the rotary axis in degrees in relation to the zero position while TRACYL is active for the first declared TRACYL transformation for each channel. | | | |
| MD irrelevant for ...... | TRACYL is not active | | | |
| Application example(s) | $MC_TRACYL_ROT_AX_OFFSET_1=15.0 | | | |
| Related to .... | TRACYL_ROT_AX_OFFSET_2 | | | |

| | | | | |
|---|---|---|---|---|
| **24808**<br>MD number | **TRACYL_DEFAULT_MODE_1**<br>Selection of TRACYL mode | | | |
| Default setting: 0 | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective afterNEWCONFIG | | Protection level: 7 / 7 | | Unit: – |
| Data type: BYTE | | Applies from SW: 7.2 | | |
| Meaning: | Selection of groove side handling method with TRACYL type 514:<br>0: Without groove side offset (for TRACYL variant 514 – corresponds to 512)<br>1: With groove side offset (for TRACYL variant 514 – corresponds to 513)<br><br>When $MC_TRAFO_TYPE_.. = 514, the selection parameter can determine whether the function is computed with or without groove side offset. The parameter defines which variant will be used if none is selected in the call parameters.<br><br>When $MC_TRAFO_DEFAULT_MODE_1 = 1, it is sufficient to specify TRACYL(30) rather than TRACYL(30,1,1) in the part program. | | | |
| MD irrelevant for ...... | TRACYL is not active | | | |
| Related to .... | TRACYL_DEFAULT_MODE_2 | | | |

| 24810<br>MD number | TRACYL_ROT_SIGN_IS_PLUS_1<br>Sign of rotary axis for the 1st TRACYL transformation | | |
|---|---|---|---|
| Default setting: 1 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: – | |
| Data type: BOOLEAN | Applies from SW: 2.0 | | |
| Meaning: | Specifies the sign which is applied to the rotary axis during the TRACYL transformation for the first declared TRACYL transformation for each channel. | | |
| MD irrelevant for ...... | TRACYL is not active | | |
| Application example(s) | $MC_TRACYL_ROT_SIGN_IS_PLUS_1=TRUE | | |
| Related to .... | TRACYL_ROT_SIGN_IS_PLUS_2 | | |

| 24820<br>MD number | TRACYL_BASE_TOOL_1[i]<br>Vector of basic tool for 1st TRACYL transformation | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: | |
| Changes effective after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: mm | |
| Data type: DOUBLE | Applies from SW: | | |
| Meaning: | MD specifies the distance of the tool zero point referred to the appropriate geometry axes valid when TRACYL is active and without tool length offset selected for the 1st TRACYL transformation for each channel.<br>Programmed length compensations are added to the base tool.<br>Index i assumes values 0, 1, 2 for the 1st to 3rd geometry axes. | | |
| MD irrelevant for ...... | TRACYL is not active | | |
| Application example(s) | $MC_TRACYL_BASE_TOOL_1[0]=tx | | |
| Related to .... | $MC_TRACYL_BASE_TOOL_2 | | |

| 24850<br>MD number | TRACYL_ROT_AX_OFFSET_2<br>Offset of rotary axis for 2nd TRACYL transformation | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: [no limit] | Maximum input limit: [no limit] | |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: Degrees | |
| Data type: DOUBLE | Applies from SW: 2.0 | | |
| Meaning: | Specifies the distance of the tool zero point offset of the rotary axis in degrees in relation to the zero position while TRACYL is active for the second declared TRACYL transformation for each channel. | | |
| MD irrelevant for ...... | TRACYL is not active | | |
| Application example(s) | $MC_TRACYL_ROT_AX_OFFSET_2=15.0 | | |
| Related to .... | TRACYL_ROT_AX_OFFSET_1 | | |

| 24858<br>MD number | TRACYL_DEFAULT_MODE_2<br>Selection of TRACYL mode | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after NEWCONFIG | Protection level: 7 / 7 | Unit: – | |
| Data type: BYTE | Applies from SW: 7.2 | | |
| Meaning: | Selection of groove side handling method with TRACYL type 514 for the 2nd TRACYL.<br>For description, see: $MC_TRAFO_DEFAULT_MODE_1 | | |
| MD irrelevant for ...... | TRACYL is not active | | |
| Related to .... | TRACYL_DEFAULT_MODE_1 | | |

| | | | | |
|---|---|---|---|---|
| **24860**<br>MD number | **TRACYL_ROT_SIGN_IS_PLUS_2**<br>Sign of rotary axis for the 2nd TRACYL transformation | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 1 |
| Changes effect. after | NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 2.0 | |
| Meaning: | Specifies the sign which is applied to the rotary axis during the TRACYL transformation for the second declared TRACYL transformation for each channel. | | | |
| MD irrelevant for ...... | TRACYL is not active | | | |
| Application example(s) | $MC_TRACYL_ROT_SIGN_IS_PLUS_2=TRUE | | | |
| Related to .... | TRACYL_ROT_SIGN_IS_PLUS_1 | | | |

| | | | | |
|---|---|---|---|---|
| **24870**<br>MD number | **TRACYL_BASE_TOOL_2[i]**<br>Vector of basic tool for 2nd TRACYL transformation | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: |
| Changes effect. after | NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | | Unit: mm |
| Data type: DOUBLE | | | Applies from SW: 2.0 | |
| Meaning: | MD specifies the distance of the tool zero point referred to the appropriate geometry axes valid when TRACYL is active and without tool length offset selected for the 2nd TRACYL transformation for each channel.<br>Programmed length compensations are added to the base tool.<br>Index i assumes values 0, 1, 2 for the 1st to 3rd geometry axes. | | | |
| MD irrelevant for ...... | TRACYL is not active | | | |
| Application example(s) | $MC_TRACYL_BASE_TOOL_2[0]=tx | | | |
| Related to .... | $MC_TRACYL_BASE_TOOL_1 | | | |

## 4.3.3 TRAANG

| | | | | | |
|---|---|---|---|---|---|
| **24700**<br>MD number | **TRAANG_ANGLE_1**<br>Angle between Cartesian axis and real (inclined) axis for the first TRAANG transformation | | | | |
| Default setting: 0 | | Minimum input limit: –90 | | Maximum input limit: 90 | |
| Changes effect. after | NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: | 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: Degrees | |
| Data type: DOUBLE | | | Applies from SW: 2.0 | | |
| Meaning: | Specifies the angle of the inclined axis in degrees between the 1st machine axis and the 1st basic axis when TRAANG is active for the first declared TRAANG transformation of the channel. The angle is counted positively in clockwise direction. | | | | |
| MD irrelevant for ...... | TRAANG is not active | | | | |
| Application example(s) | $MC_TRAANG_ANGLE_1=15.0 | | | | |
| Related to .... | TRAANG_ANGLE_2 | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **24710**<br>MD number | **TRAANG_BASE_TOOL_1[i]**<br>Vector of base tool for first TRAANG transformation [axis no.]: 0 ... 2 | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 2 | |
| Changes effect. after | NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: | 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: mm | |
| Data type: DOUBLE | | | Applies from SW: 2.0 | | |
| Meaning: | MD specifies the distance of the tool zero point referred to the appropriate geometry axes valid with TRAANG active and without tool length offset selected for the 1st TRAANG transformation for each channel.<br>Programmed length compensations are added to the base tool.<br>Index i assumes values 0, 1, 2 for the 1st to 3rd geometry axes. | | | | |
| MD irrelevant for ...... | | | | | |
| Application example(s) | $MC_TRAANG_BASE_TOOL_1[0]=tx | | | | |
| Related to .... | $MC_TRAANG_BASE_TOOL_2 | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **24720**<br>MD number | **TRAANG_PARALLEL_VELO_RES_1** | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effect. after | NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: | 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: – | |
| Data type: DOUBLE | | | Applies from SW: 2.0 | | |
| Meaning: | Specifies the velocity reserve for jog, positioning and oscillation movements which is held ready on the parallel axis (see $MC_TRAFO_AXES_IN_n[1]) for the compensatory movement; MD setting applies to the first TRAANG transformation for each channel. | | | | |
| MD irrelevant for ...... | TRAANG is not active | | | | |
| Application example(s) | $MC_TRAANG_PARALLEL_VELO_RES_1=0 | | | | |
| Related to .... | TRAANG_PARALLEL_VELO_RES_2 | | | | |

| | | | |
|---|---|---|---|
| **24721**<br>MD number | **TRAANG_PARALLEL_VELO_RES_2** | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: – | |
| Data type: DOUBLE | Applies from SW: 2.0 | | |
| Meaning: | Specifies the velocity reserve for jog, positioning and oscillation movements which is held ready on the parallel axis (see $MC_TRAFO_AXES_IN_n[1]) for the compensatory movement; MD setting applies to the second TRAANG transformation for each channel. | | |
| Application example(s) | $MC_TRAANG_PARALLEL_VELO_RES_2=0 | | |
| Related to .... | $MC_TRAANG_PARALLEL_VELO_RES_1 | | |

| | | | |
|---|---|---|---|
| **24750**<br>MD number | **TRAANG_ANGLE_2**<br>Angle between Cartesian axis and real (inclined) axis for second TRAANG transformation | | |
| Default setting: 0 | Minimum input limit: –90 | Maximum input limit: 90 | |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: Degrees | |
| Data type: DOUBLE | Applies from SW: 2.0 | | |
| Meaning: | Specifies the angle of the inclined axis in degrees between the 1st machine axis and the 1st basic axis when TRAANG is active for the second declared TRAANG transformation of the channel. The angle is counted positively in clockwise direction. | | |
| MD irrelevant for ...... | TRAANG is not active | | |
| Application example(s) | $MC_TRAANG_ANGLE_1=15.0 | | |
| Related to .... | TRAANG_ANGLE_1 | | |

| | | | |
|---|---|---|---|
| **24760**<br>MD number | **TRAANG_BASE_TOOL_2[i]**<br>Vector of base tool for second TRAANG transformation [axis no.]: 0 ... 2 | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 2 | |
| Changes effective after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: mm | |
| Data type: DOUBLE | Applies from SW: 2.0 | | |
| Meaning: | MD specifies the distance of the tool zero point referred to the appropriate geometry axes valid with TRAANG active and without tool length offset selected for the 2nd TRAANG transformation for each channel.<br>Programmed length compensations are added to the base tool.<br>Index i assumes values 0, 1, 2 for the 1st to 3rd geometry axes. | | |
| MD irrelevant for ...... | TRAANG is not active | | |
| Application example(s) | $MC_TRAANG_BASE_TOOL_2[0]=tx | | |
| Related to .... | $MC_TRAANG_BASE_TOOL_1 | | |

| 24770<br>MD number | TRAANG_PARALLEL_ACCEL_RES_1 | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effect. after: NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: – | |
| Data type: DOUBLE | Applies from SW: 2.0 | | |
| Meaning: | Specifies the axis acceleration reserve for jog, positioning and oscillation movements which is held ready on the parallel axis (see $MC_TRAFO_AXES_IN_n[1]) for the compensatory movement; MD setting applies to the first TRAANG transformation for each channel. | | |
| MD irrelevant for ...... | TRAANG is not active | | |
| Application example(s) | $MC_TRAANG_PARALLEL_ACCEL_RES_1=0 | | |
| Related to .... | TRAANG_PARALLEL_ACCEL_RES_2 | | |

| 24771<br>MD number | TRAANG_PARALLEL_ACCEL_RES_2 | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effect. after: NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: – | |
| Data type: DOUBLE | Applies from SW: 2.0 | | |
| Meaning: | Specifies the axis acceleration reserve for jog, positioning and oscillation movements which is held ready on the parallel axis (see $MC_TRAFO_AXES_IN_n[1]) for the compensatory movement; MD setting applies to the second TRAANG transformation for each channel. | | |
| MD irrelevant for ...... | TRAANG is not active | | |
| Application example(s) | $MC_TRAANG_PARALLEL_RES_2=0 | | |
| Related to .... | $MC_TRAANG_PARALLEL_RES_1 | | |

## 4.3.4 MD for chained transformations

| | | | | | |
|---|---|---|---|---|---|
| **24995**<br>MD number | **TRACON_CHAIN_1[n]**<br>Transformation chain of the first chained transformation | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 8 | |
| Changes effect. after | NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protect. level: | 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: – | |
| Data type: DWORD | | | Applies from SW: 5 | | |
| Meaning: | The MD is saved internally as a table. In the table the numbers of the transformations to be chained are specified in the same sequence as the transformations are to be implemented from the BCS to the MCS. n stands for the index of entries in the MD.<br><br>Example:<br>Optionally, a machine can be operated as a 5-axis machine or as a Transmit machine. A linear axis is not perpendicular to the other linear axes (inclined axis).<br>5 transformations must be set via machine data, e.g.<br><br>TRAFO_TYPE_1 = 16 (5-axis trafo), first transformation<br>TRAFO_TYPE_2 = 256 (Transmit), second transformation<br>TRAFO_TYPE_3 = 1024 (Inclined axis), third transformation<br>TRAFO_TYPE_4 = 8192 First chained transformation, fourth transformation<br>TRAFO_TYPE_5 = 8192 Second chained transformation, fifth transformation<br><br>If the 4th transformation is to be the chaining: 5-axis transformation / inclined axis and the 5th transformation is to be the chaining: Transmit / inclined axis,<br>then TRACON_CHAIN_1 (1, 3, 0, 0) is entered in the first table and TRACON_CHAIN_2 (2, 3, 0, 0) in the second table. Detailed notation shown in the example in Chapter 6.<br>Entry 0 means no transformation (a 3rd and 4th transformation cannot be chained in SW 5).<br>The transformations can be assigned (TRAFO_TYPE_1 to TRAFO_TYPE_10) in any sequence. The chained transformations do not have to be the last ones. However, they must be behind all transformations that occur in a transformation chain. In the preceding example, this would mean that, for example, the position of the third and fourth transformation must not be swapped. It would be possible though to define a sixth transformation if it is not to be included in a chained transformation.<br>However, not just any transformation can be chained to another one.<br>The following restrictions apply in SW version 5:<br>• The first transformation in the chain must be:<br>– An orientation transformation (3, 4, 5-axis transformation, universal milling head),<br>– Transmit or<br>– Peripheral transformation or<br>– Inclined axis<br>• The second transformation must be an inclined axis transformation.<br>• Only two transformations may be chained.<br>It is permissible (e.g. for test purposes) to enter only one transformation in the list. | | | | |
| MD irrelevant for ...... | TRAFOOF | | | | |
| Application example(s) | Chapter 6 | | | | |
| Special cases, errors, ...... | More than 2 transformations in the chain, 2nd transformation not TRAANG | | | | |
| Related to .... | MD 24100: TRAFO_TYPE | | | | |
| References | /FB/, F2, “3 to 5 Axis Transformation” | | | | |

| | | | |
|---|---|---|---|
| **24996**<br>MD number | **TRACON_CHAIN_2[n]**<br>Transformation chain of the second chained transformation | | |
| Default setting: | Minimum input limit: 0 | Maximum input limit: 8 extended to 10 | |
| Changes effect. after NEWCONFIG (SW 5.2 and higher)<br>POWER ON (SW 5.1 and lower) | Protection level: 7 / 7 (SW 5.2 and higher)<br>2 / 7 (up to SW 5.1) | Unit: – | |
| Data type: DWORD | Applies from SW: 5 | | |
| Meaning: | Analogous to TRACON_CHAIN_1, but for the second chained transformation in the channel | | |
| MD irrelevant for ...... | TRAFOOF | | |
| Application example(s) | Chapter 6 | | |
| Special cases, errors, ...... | More than 2 transformations in the chain, 2nd transformation not TRAANG | | |
| Related to .... | MD 24100: TRAFO_TYPE | | |
| References | /FB/, F2, “3 to 5 Axis Transformation” | | |

| | | | |
|---|---|---|---|
| **24997**<br>MD number | **TRACON_CHAIN_3[n]**<br>Transformation chain of the third chained transformation | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 10 | |
| Changes effective afterNEWCONFIG | Protection level: 7 / 7 | Unit: – | |
| Data type: DWORD | Applies from SW: 7.2 | | |
| Meaning: | Analogous to TRACON_CHAIN_1, but for the third chained transformation in the channel | | |
| MD irrelevant for ...... | TRAFOOF | | |
| Application example(s) | Chapter 6 | | |
| Special cases, errors, ...... | More than 2 transformations in the chain, 2nd transformation not TRAANG | | |
| Related to .... | MD 24100: TRAFO_TYPE | | |
| References | /FB/, F2, “3 to 5 Axis Transformation” | | |

| | | | |
|---|---|---|---|
| **24998**<br>MD number | **TRACON_CHAIN_4[n]**<br>Transformation chain of the fourth chained transformation | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 10 | |
| Changes effective afterNEWCONFIG | Protection level: 7 / 7 | Unit: – | |
| Data type: DWORD | Applies from SW: 7.2 | | |
| Meaning: | Analogous to TRACON_CHAIN_1, but for the fourth chained transformation in the channel | | |
| MD irrelevant for ...... | TRAFOOF | | |
| Application example(s) | Chapter 6 | | |
| Special cases, errors, ...... | More than 2 transformations in the chain, 2nd transformation not TRAANG | | |
| Related to .... | MD 24100: TRAFO_TYPE | | |
| References | /FB/, F2, “3 to 5 Axis Transformation” | | |

■

# Notes

# Signal Descriptions 5

## 5.1 TRANSMIT

| DB 21, ...<br>**DBX 33.6**<br>Data Block | **Transformation active**<br><br>Signal(s) from NCK channel (NCK–>PLC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.2 |
| Signal state 1 or signal transition 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 signal transition 1 ––––> 0 | No transformation is active. | |
| References | /PA1/, "Programming Guide"<br>/FB/, F2, "5-Axis Transformation" | |

## 5.2 TRACYL

See 5.1.

## 5.3 TRAANG

See 5.1.

■

# Notes

# Example 6

## 6.1 TRANSMIT

The following programming example relates to the configuration illustrated in Fig. 6-1 and shows the sequence of main steps required to configure the axes and activate TRANSMIT.

;General axis configuration for turning

```
$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"       ; Geometry axis
$MC_AXCONF_GEOAX_NAME_TAB[1] = "Y"       ; Geometry axis
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"       ; Geometry axis
$MC_AXCONF_GEOAX_ASSIGN_TAB[0] =1        ; X as channel axis 1
$MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0       ; Y is not a channel axis
$MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 2       ; Z as channel axis 2
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XC";
$MC_AXCONF_CHANAX_NAME_TAB[1] = "ZC";
$MC_AXCONF_CHANAX_NAME_TAB[2] = "CC";
$MC_AXCONF_CHANAX_NAME_TAB[3] = "ASC";
$MC_AXCONF_CHANAX_NAME_TAB[4] = "";
$MC_AXCONF_MACHAX_USED[0] = 2            ; XC as machine axis 2
$MC_AXCONF_MACHAX_USED[1] = 3            ; ZC as machine axis 3
$MC_AXCONF_MACHAX_USED[2] = 1            ; CC as machine axis 1
$MC_AXCONF_MACHAX_USED[3] = 4            ; ASC as machine axis  4
$MC_AXCONF_MACHAX_USED[3] = 0            ; Empty
$MA_SPIND_ASSIGN_TO_MACHAX[AX1]= 1 ; C is spindle 1
$MA_SPIND_ASSIGN_TO_MACHAX[AX2]= 0 ; X is not a spindle
$MA_SPIND_ASSIGN_TO_MACHAX[AX3]= 0 ; Z is not a spindle
$MA_SPIND_ASSIGN_TO_MACHAX[AX4]= 2 ; AS is spindle 2
$MN_AXCONF_MACHAX_NAME_TAB[0]= "CM" ; 1st machine axis
$MN_AXCONF_MACHAX_NAME_TAB[1]= "XM" ; 2nd machine axis
$MN_AXCONF_MACHAX_NAME_TAB[2]= "ZM" ; 3rd machine axis
$MN_AXCONF_MACHAX_NAME_TAB[3]= "ASM" ; 4th machine axis
```

```
;Prepare for TRANSMIT (as first and only transformation)

$MA_ROT_IS_MODULO[3] = TRUE ; c as modulo axis
$MC_TRAFO_TYPE_1 = 256 ; TRANSMIT transformation
$MC_TRAFO_AXES_IN_1[0] = 1 ; Channel axis perpendicular to rotary axis
$MC_TRAFO_AXES_IN_1[1] = 3 ; Channel axis rotary axis
$MC_TRAFO_AXES_IN_1[2] = 2 ; Channel axis parallel to rotary axis
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1  ; 1st channel axis becomes
                                      : GEOAX X
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 3  ; 2nd channel axis becomes
                                      ; GEOAX Y
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 2  ; 3rd channel axis becomes
                                      ; GEOAX Z
$MC_TRANSMIT_ROT_AX_OFFSET_1 = 0. ; Rotational position X-Y plane
                                  ; in rel. to zero of rotary axis
$MC_TRANSMIT_ROT_SIGN_IS_PLUS_1 = FALSE ; Rotary axis turns –
$MC_TRANSMIT_BASE_TOOL_1 [0] = 0.0 ; Tool distance in X
$MC_TRANSMIT_BASE_TOOL_1 [1] = 0.0 ; Tool distance in Y
$MC_TRANSMIT_BASE_TOOL_1 [2] = 0.0 ; Tool distance in Z


; Activation of
TRANSMIT

; Programming in X,Y, Z

; Return to rotational operation
TRAFOOF
```

TRACYL

The following programming example relates to the configuration illustrated in Fig. 6-1 and shows the sequence of main steps required to configure the axes and activate TRACYL.

```
; General axis configuration for turning

$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"        ; Geometry axis
$MC_AXCONF_GEOAX_NAME_TAB[1] = "Y"        ; Geometry axis
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"        ; Geometry axis
$MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1        ; X as channel axis 1
$MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2        ; Y is not a channel axis
$MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3        ; Z as channel axis 2
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XC";
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YC";
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZC";
$MC_AXCONF_CHANAX_NAME_TAB[3] = "CC";
$MC_AXCONF_CHANAX_NAME_TAB[4] = "ASC";
$MC_AXCONF_MACHAX_USED[0] = 2     ; X as machine axis 2
$MC_AXCONF_MACHAX_USED[1] = 3     ; Y as machine axis 3
$MC_AXCONF_MACHAX_USED[2] = 4     ; Z as machine axis 4
$MC_AXCONF_MACHAX_USED[3] = 1     ; C as machine axis 1
$MC_AXCONF_MACHAX_USED[4] = 5     ; AS as machine axis 5
$MA_SPIND_ASSIGN_TO_MACHAX[AX1]= 1 ; C is spindle 1
$MA_SPIND_ASSIGN_TO_MACHAX[AX2]= 0 ; X is not a spindle
$MA_SPIND_ASSIGN_TO_MACHAX[AX3]= 0 ; Y is not a spindle
$MA_SPIND_ASSIGN_TO_MACHAX[AX4]= 0 ; Z is not a spindle
$MA_SPIND_ASSIGN_TO_MACHAX[AX5]= 2 ; AS is spindle 2
$MN_AXCONF_MACHAX_NAME_TAB[0]= "CM" ; 1st machine axis
$MN_AXCONF_MACHAX_NAME_TAB[1]= "XM" ; 2nd machine axis
$MN_AXCONF_MACHAX_NAME_TAB[2]= "YM" ; 3rd machine axis
```

$MN_AXCONF_MACHAX_NAME_TAB[3]= “ZM” ; 4th machine axis
$MN_AXCONF_MACHAX_NAME_TAB[4]= “ASM” ; 5th machine axis

;Prepare for TRACYL (first and only transformation)

$MC_TRAFO_TYPE_1 = 513 ; TRACYL transformation with groove wall compensation
$MC_TRAFO_AXES_IN_1[0] = 1 ; Channel axis radial to rotary axis
$MC_TRAFO_AXES_IN_1[1] = 4 ; Channel axis in cylinder generated surface perpendicular to rotary axis
$MC_TRAFO_AXES_IN_1[2] = 3 ; Channel axis parallel to rotary axis
$MC_TRAFO_AXES_IN_1[3] = 2 ; Channel axis is add. axis for index [0]

$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1 ; 1st channel axis becomes ; GEOAX X
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 4 ; 2nd channel axis becomes GEOAX Y $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3 ; 3rd channel axis ; becomes GEOAX Z

$MC_TRACYL_ROT_AX_OFFSET_1 = 0. ; Rotational position X-Y plane in rel. to zero pos. of rotary axis
$MC_TRACYL_ROT_SIGN_IS_PLUS_1 = FALSE ; Rotary axis turns –
$MC_TRACYL_BASE_TOOL_1 [0] = 0.0 ; Tool distance in X
$MC_TRACYL_BASE_TOOL_1 [1] = 0.0 ; Tool distance in Y
$MC_TRACYL_BASE_TOOL_1 [2] = 0.0 ; Tool distance in Z

; Activation of
TRACYL(40.0)

; See below for programming in Y and Z

; Return to rotational operation
TRAFOOF

**Programming with groove wall compensation**

(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 Fig. 6-1).

**Tool radius**

The tool radius is automatically taken into account with respect to the groove side wall (see Fig. 6-1). 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).

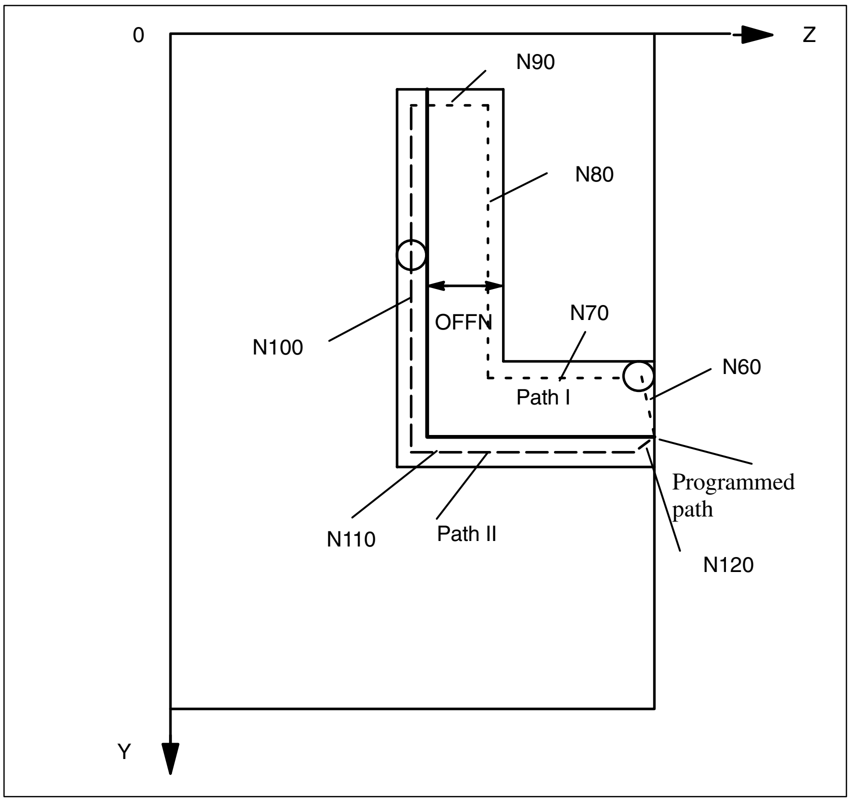



Fig. 6-1 Groove with wall compensation, cylinder coordinates (simplified sketch)

; Example program which leads the tool along path I across path II back to the
; initial position after transformation selection (machine data see Chapter 4,
; example X-Y-Z-C kinematics):

```
N1   SPOS=0;                  Transfer of spindle to rotary axis mode
N5    G0 X25 Y0 Z105 CC=200 F5000 G64;
                              Positioning of machine above groove center
N10 TRACYL(40.);              Transformation selection with
                              ; reference diameter 40 mm
N20  G19 G90;                 Machining plane is generated cylinder surface
N30   T1 D1;                  Tool selection, can also be positioned before
                              TRACY (..)
N40   G1 X20;                 Infeed tool to groove base
N50   OFFN=12.;               Define groove wall distance, must not be
                              ; in a separate line

; Approach groove wall
N60 G1 Z100 G42;              TRC selection to approach groove wall

; Machining of groove section path I
N70 G1 Z50;                   Groove section parallel to cylinder plane
N80 G1 Y10;                   Groove section parallel to periphery
```

```
                           ; Approach groove wall for path II
N90 OFFN=4 G42;            Define groove wall distance and
                           ; TRC selection to approach groove wall

; Machine groove section path II
N100 G1 Y70;               Corresponds to CC=200 degrees
N110 G1 Z100;              Back to initial value

; Retract from groove wall
N120 G1 Z105 G40;          TRC deselection to retract from groove wall
N130 G0 X25;               Retract from groove

N140 TRAFOOF;
N150 G0 X25 Y0 Z105 CC=200 D0;
                           Return to initial point and
                           ; deselect tool compensation
N160 M30
```

**Programming without groove side offset**

TRACYL without groove side offset, with additional linear axis

(TRAFO_TYPE_n=513)

; The following part program requires the machine data settings shown below:

```
$MC_AXCONF_MACHAX_USED[0] = 1      ;X as machine axis 1
$MC_AXCONF_MACHAX_USED[1] = 2      ;Y as machine axis 2
$MC_AXCONF_MACHAX_USED[2] = 3      ;Z as machine axis 3
$MC_AXCONF_MACHAX_USED[3] = 4      ;C as machine axis 4
$MC_AXCONF_CHANAX_NAME_TAB[1] = "Y2"

$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1       ; X as channel axis 1
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 2       ; Y not a channel axis
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3       ; Z as channel axis 2

$MC_TRAFO_TYPE_1 = 514 ; TRACYL without groove side offset, with
                         extended tool length compensation

$MC_TRAFO_AXES_IN_1[0] = 1 ; channel axis radial to rotary axis
$MC_TRAFO_AXES_IN_1[1] = 4 ; channel axis in cylinder generated surface,
                             perpendicular to rotary axis
$MC_TRAFO_AXES_IN_1[2] = 3 ; channel axis parallel to rotary axis
$MC_TRAFO_AXES_IN_1[3] = 2 ; channel axis additional axis to index [0]

$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1 ; 1st channel axis will be
                                       GEOAX X
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 4 ; 2nd channel axis will be
                                       GEOAX Y
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3 ; 3rd channel axis will be
                                       GEOAX Z

$MC_TRACYL DEFAULT MODE_1 = 0              ; or not set at all
```

**Tool data:**

```
$TC_DP1[1,1]=120                ; tool type end mill
$TC_DP2[1,1]=0
$TC_DP3[1,1]=                   ; length compensation vector
$TC_DP4[1,1]=25
$TC_DP5[1,1]=5
$TC_DP6[1,1]=4                  ; radius, tool radius
```

**Part program:**

```
N1001 T1 D1 G54 G19 G90 F5000 G64
N1005 G0 X25 Y0 Z105 A=200

                       ; Selection of 1st TRACYL without groove side offset
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 1st TRACYL with groove side offset
N2010 G0 TRACYL(40.,1,1) ; TRCYL(40., ,1) could also be selected

N2040 G1 X20;
N2060 G1 Z100;
N2070 G1 Z50;
N2080 G1 Y10;

N2140 TROFOOF
```

## 6.2 TRAANG (inclined axis)

The following programming example relates to the configuration illustrated in Fig. 6-1 and shows the sequence of main steps required to configure the axes and activate TRAANG.

;General axis configuration for grinding

```
$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"       ; Geometry axis
$MC_AXCONF_GEOAX_NAME_TAB[1] = ""        ; Geometry axis
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"       ; Geometry axis
$MC_AXCONF_GEOAX_ASSIGN_TAB[0] =0        ; X is not a channel axis
$MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0       ; Y is not a channel axis
$MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 1       ; Z as channel axis 1
$MC_AXCONF_CHANAX_NAME_TAB[0] = "Z";
$MC_AXCONF_CHANAX_NAME_TAB[1] = "C";
$MC_AXCONF_CHANAX_NAME_TAB[2] = "AS1";
$MC_AXCONF_CHANAX_NAME_TAB[3] = "MU"
$MC_AXCONF_MACHAX_USED[0] = 3            ; Z as machine axis 3
$MC_AXCONF_MACHAX_USED[1] = 1            ; C as machine axis 1
$MC_AXCONF_MACHAX_USED[2] = 4            ; AS as machine axis 4
```

```
$MC_AXCONF_MACHAX_USED[3] = 2                ; MU as machine axis 2
$MC_AXCONF_MACHAX_USED[3] = 0                ; Empty
$MC_AXCONF_MACHAX_USED[3] = 0                ; Empty
$MA_SPIND_ASSIGN_TO_MACHAX[AX1] = 1 ; C is spindle 1
$MA_SPIND_ASSIGN_TO_MACHAX[AX2] = 0 ; X is not a spindle
$MA_SPIND_ASSIGN_TO_MACHAX[AX3] = 0 ; Z is not a spindle
$MA_SPIND_ASSIGN_TO_MACHAX[AX4] = 2 ; AS is spindle 2
$MN_AXCONF_MACHAX_NAME_TAB[0] = "C1" ; 1st machine axis
$MN_AXCONF_MACHAX_NAME_TAB[1] = "MU" ; 2nd machine axis
$MN_AXCONF_MACHAX_NAME_TAB[2] = "MZ" ; 3rd machine axis
$MN_AXCONF_MACHAX_NAME_TAB[3] = "AS1" ; 4th machine axis

;Prepare for TRAANG (first and only transformation)

$MC_TRAFO_TYPE_1 = 1024 ; TRAANG transformation
$MC_TRAFO_AXES_IN_1[0] = 4 ; Channel axis "inclined axis"
$MC_TRAFO_AXES_IN_1[1] = 1 ; Channel axis parallel to axis Z
$MC_TRAFO_AXES_IN_1[2] = 0 ; Channel axis not active
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 4 ; X 1st channel axis
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 0 ; Y 2nd channel axis
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 1 ; Z 3rd channel axis
$MC_TRAANG_ANGLE_1 = 30. ; Angle of inclined axis
$MC_TRAANG_BASE_TOOL_1 [0] = 0 ; Tool distance in X
$MC_TRAANG_BASE_TOOL_1 [1] = 0 ; Tool distance in Y
$MC_TRAANG_BASE_TOOL_1 [2] = 0 ; Tool distance in Z

TRAANG        ; Activation

              ; Programming in X,Y, Z

TRAFOOF       ; Return to turning mode
```

## 6.3 Chained transformations

**Examples**

The following elements are determined in the next chapter:

- 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:

1. 5-axis transformation with rotatable tool and axis sequence AB (trafo type 16)
2. Transmit (trafo type 256)
3. Inclined axis (trafo type 1024)
4. Chaining of the 1st and 3rd transformation (trafo type 8192)
5. Chaining of the 2nd and 3rd transformation (trafo type 8192)

**General channel configuration**

```
CHANDATA (1)    ; Channel data in channel 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] = 7
$MC_AXCONF_MACHAX_USED[7] = 0

$MC_AXCONF_CHANAX_NAME_TAB[3] = “A”
$MC_AXCONF_CHANAX_NAME_TAB[4] = “B”
$MC_AXCONF_CHANAX_NAME_TAB[5] = “C”

$MA_IS_ROT_AX[ AX4 ] = TRUE
$MA_IS_ROT_AX[ AX5 ] = TRUE
$MA_IS_ROT_AX[ AX6 ] = TRUE
$MA_IS_ROT_AX[ AX7 ] = TRUE

$MA_SPIND_ASSIGN_TO_MACHAX[ AX5 ] = 0
$MA_SPIND_ASSIGN_TO_MACHAX[AX7] = 1
$MA_ROT_IS_MODULO[AX7] = TRUE
```

**Single transformations**

```
; 1st TRAORI
$MC_TRAFO_TYPE_1= 16          ; TRAORI: A-B kinematics
$MC_TRAFO_AXES_IN_1[0]=1
$MC_TRAFO_AXES_IN_1[1]=2
$MC_TRAFO_AXES_IN_1[2]=3
$MC_TRAFO_AXES_IN_1[3]=4
$MC_TRAFO_AXES_IN_1[4]=5
$MC_TRAFO_AXES_IN_1[5]=0
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3
$MC_TRAFO5_BASE_TOOL_1[0]=0
$MC_TRAFO5_BASE_TOOL_1[1]=0
$MC_TRAFO5_BASE_TOOL_1[2]=0

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

; 3. TRAANG
$MC_TRAFO_TYPE_3=1024 ;TRAANG
$MC_TRAFO_AXES_IN_3[0]=1
$MC_TRAFO_AXES_IN_3[1]=3
$MC_TRAFO_AXES_IN_3[2]=2
$MC_TRAFO_AXES_IN_3[3]=0
$MC_TRAFO_AXES_IN_3[4]=0
$MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] = 1
$MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] = 3
$MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] = 2
$MC_TRAANG_ANGLE_1=45.
$MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
$MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
$MC_TRAANG_BASE_TOOL_1[0] = 0.0
$MC_TRAANG_BASE_TOOL_1[1] = 0.0
$MC_TRAANG_BASE_TOOL_1[2] = 0.0
```

**Chained transformations**

```
; 4th TRACON (chaining of TRAORI / TRAANG)
$MC_TRAFO_TYPE_4 = 8192

$MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] = 2
$MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] = 1
$MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] = 3

$MC_TRACON_CHAIN_1[0] = 1
$MC_TRACON_CHAIN_1[1] = 3
$MC_TRACON_CHAIN_1[2] = 0

; 5. TRACON (chaining of TRANSMIT / TRAANG)

$MC_TRAFO_TYPE_5 = 8192

$MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] = 1
$MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] = 6
$MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] = 3

$MC_TRACON_CHAIN_2[0] = 2
$MC_TRACON_CHAIN_2[1] = 3
$MC_TRACON_CHAIN_2[2] = 0
```

**Parts program (extracts)**

Example of an NC program which uses the set transformations:

**; Call single transformations**

```
                        ; Tool definition
$TC_DP1[1,1]=120        ; Tool type
$TC_DP3[1,1]= 10        ; Tool length

n2 x0 y0 z0 a0 b0 f20000 t1 d1
n4 x20

n30 TRANSMIT                      ; Activate Transmit
n40 x0 y20
n50 x–20 y0
n60 x0 y–20
n70 x20 y0
n80 TRAFOOF                       ; Deactivate Transmit


n130 TRAANG(45.)        ; Activate inclined axis transformation, parameter: Angle 45°
n140 x0 y0 z20
n150 x–20 z0
n160 x0 z–20
n170 x20 z0
```

---

**Note**

The above examples assume that the angle of the “inclined axis” can be set on the machine and is set to 0 degrees when the single transformation is activated.

---

```
                              ; 1. Call chained transformation
                              ; TRAORI + TRAANG

n230 TRACON(1, 45.)     ; Activate 1st of 2 chained transformations
                        ; 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
...


                              ; 2. Call 2nd chained transformation
                              ; TRANSMIT + TRAANG

n230 TRACON(2, 40.)     ; Activate 2nd 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                  ; Deactivate 2nd chained transformation
...

n1000 M30
```

## 6.4 Activating transformation MD via a parts program (SW 5.2 and higher)

It would be permissible in the following example to reconfigure (write) a machine data affecting the second transformation (e.g. $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:**

```
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 an MD
                                        ; of the 1st orientation
                                        ; transformation
N100 A20
N110 X20
N120 X0
N130 NEWCONF                            ; Accept newly modified
                                        ; machine data
N140 TRAORI(1)                          ; Select 1st orientation
                                        ; transformation,
                                        ; MD becomes operative
N150 G19 X0 Y0 Z0
N160 A50 B50
N170 A0 B0
N180 TRAFOOF
N190 M30
```

■

# 7 Data Fields, Lists

## 7.1 TRANSMIT

### 7.1.1 Interface signals

| DB number | Bit, byte | Name | Refer-ence |
|---|---|---|---|
| **Channel-specific** | | | |
| 21, ... | 33.6 | Transformation active | F2 |

### 7.1.2 Machine data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Channel-specific ($MC_...)** | | | |
| 20110 | RESET_MODE_MASK | Definition of control basic setting after run-up and RESET/part program end | K2 |
| 20140 | TRAFO_RESET_VALUE | Basic transformation position | K2 |
| 22534 | TRAFO_CHANGE_M_CODE | M code for transformation changeover | |
| 24100 | TRAFO_TYPE_1 | Definition of the 1st transformation in channel | F2 |
| 24110 | TRAFO_AXES_IN_1 | Axis assignment for the 1st transformation | F2 |
| 24120 | TRAFO_GEOAX_ASSIGN_TAB_1 | Geo-axis assignment for 1st transformation | F2 |
| 24200 | TRAFO_TYPE_2 | Definition of the 2nd transformation in channel | F2 |
| 24210 | TRAFO_AXES_IN_2 | Axis assignment for the 2nd transformation | F2 |
| 24220 | TRAFO_GEOAX_ASSIGN_TAB_2 | Geo-axis assignment for 2nd transformation | F2 |

| Number | Names | Name | Reference |
|---|---|---|---|
| 24300 | TRAFO_TYPE_3 | Definition of the 3rd transformation in channel | F2 |
| 24310 | TRAFO_AXES_IN_3 | Axis assignment for the 3rd transformation | F2 |
| 24320 | TRAFO_GEOAX_ASSIGN_TAB_3 | Geo-axis assignment for 3rd transformation | F2 |
| 24400 | TRAFO_TYPE_4 | Definition of the 4th transformation in channel | F2 |
| 24410 | TRAFO_AXES_IN_4 | Axis assignment for the 4th transformation | F2 |
| 24420 | TRAFO_GEOAX_ASSIGN_TAB_4 | Geo-axis assignment for 4th transformation | F2 |
| 24430 | TRAFO_TYPE_5 | Definition of the 5th transformation in channel | F2 |
| 24432 | TRAFO_AXES_IN_5 | Axis assignment for the 5th transformation | F2 |
| 24434 | TRAFO_GEOAX_ASSIGN_TAB_5 | Geo-axis assignment for 5th transformation | F2 |
| 24440 | TRAFO_TYPE_6 | Definition of the 6th transformation in channel | F2 |
| 24442 | TRAFO_AXES_IN_6 | Axis assignment for the 6th transformation | F2 |
| 24444 | TRAFO_GEOAX_ASSIGN_TAB_6 | Geo-axis assignment for 6th transformation | F2 |
| 24450 | TRAFO_TYPE_7 | Definition of the 7th transformation in channel | F2 |
| 24452 | TRAFO_AXES_IN_7 | Axis assignment for the 7th transformation | F2 |
| 24454 | TRAFO_GEOAX_ASSIGN_TAB_7 | Geo-axis assignment for 7th transformation | F2 |
| 24460 | TRAFO_TYPE_8 | Definition of the 8th transformation in channel | F2 |
| 24462 | TRAFO_AXES_IN_8 | Axis assignment for the 8th transformation | F2 |
| 24464 | TRAFO_GEOAX_ASSIGN_TAB_8 | Geo-axis assignment for 8th transformation | F2 |
| 24900 | TRANSMIT_ROT_AX_OFFSET_1 | Deviation of rotary axis from zero position in degrees (1st TRANSMIT) | |
| 24910 | TRANSMIT_ROT_SIGN_IS_PLUS_1 | Sign of rotary axis for TRANSMIT (1st TRANSMIT) | |
| 24911 | TRANSMIT_POLE_SIDE_FIX_1 | Limitation of working range in front of/behind pole, 1st transformation | |
| 24920 | TRANSMIT_BASE_TOOL_1 | Distance of tool zero point from origin of geo-axes (1st TRANSMIT) | |
| 24950 | TRANSMIT_ROT_AX_OFFSET_2 | Deviation of rotary axis from zero position in degrees (2nd TRANSMIT) | |
| 24960 | TRANSMIT_ROT_SIGN_IS_PLUS_2 | Sign of rotary axis for TRANSMIT (2nd TRANSMIT) | |
| 24961 | TRANSMIT_POLE_SIDE_FIX_2 | Limitation of working range in front of/behind pole, 2nd transformation | |
| 24970 | TRANSMIT_BASE_TOOL_2 | Distance of tool zero point from origin of geo-axes (2nd TRANSMIT) | |

## 7.1.3 Interrupts

The alarms which may occur in conjunction with the TRANSMIT transformation are explained in the Diagnostic Guide and in the online help in systems with MMC 101/102/103

**References:** /DA/, “Diagnostic Guide”

# 7.2 TRACYL

## 7.2.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **Channel-specific** | | | |
| 21, ... | 33.6 | Transformation active | F2 |

## 7.2.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channel-specific ($MC_...)** | | | |
| 20110 | RESET_MODE_MASK | Definition of control basic setting after run-up and RESET/part program end | K2 |
| 20140 | TRAFO_RESET_VALUE | Basic transformation position | K2 |
| 20144 | TRAFO_MODE_MASK | Selection of the kinematic transformation function | |
| 24100 | TRAFO_TYPE_1 | Definition of the 1st transformation in channel | F2 |
| 24110 | TRAFO_AXES_IN_1 | Axis assignment for the 1st transformation | F2 |
| 24120 | TRAFO_GEOAX_ASSIGN_TAB_1 | Geo-axis assignment for 1st transformation | F2 |
| 24130 | TRAFO_INCLUDES_TOOL_1 | Tool handling with active transformation 1. | F2 |
| 24200 | TRAFO_TYPE_2 | Definition of the 2nd transformation in channel | F2 |
| 24210 | TRAFO_AXES_IN_2 | Axis assignment for the 2nd transformation | F2 |
| 24220 | TRAFO_GEOAX_ASSIGN_TAB_2 | Geo-axis assignment for 2nd transformation | F2 |
| 24230 | TRAFO_INCLUDES_TOOL_2 | Tool handling with active transformation 2. | F2 |
| 24300 | TRAFO_TYPE_3 | Definition of the 3rd transformation in channel | F2 |
| 24310 | TRAFO_AXES_IN_3 | Axis assignment for the 3rd transformation | F2 |
| 24320 | TRAFO_GEOAX_ASSIGN_TAB_3 | Geo-axis assignment for 3rd transformation | F2 |
| 24330 | TRAFO_INCLUDES_TOOL_3 | Tool handling with active transformation 3. | F2 |
| 24400 | TRAFO_TYPE_4 | Definition of the 4th transformation in channel | F2 |
| 24410 | TRAFO_AXES_IN_4 | Axis assignment for the 4th transformation | F2 |
| 24420 | TRAFO_GEOAX_ASSIGN_TAB_4 | Geo-axis assignment for 4th transformation | F2 |
| 24426 | TRAFO_INCLUDES_TOOL_4 | Tool handling with active transformation 4. | F2 |
| 24430 | TRAFO_TYPE_5 | Definition of the 5th transformation in channel | F2 |
| 24432 | TRAFO_AXES_IN_5 | Axis assignment for the 5th transformation | F2 |

| Number | Names | Name | Reference |
|---|---|---|---|
| 24434 | TRAFO_GEOAX_ASSIGN_TAB_5 | Geo-axis assignment for 5th transformation | F2 |
| 24436 | TRAFO_INCLUDES_TOOL_5 | Tool handling with active transformation 5 | F2 |
| 24440 | TRAFO_TYPE_6 | Definition of the 6th transformation in channel | F2 |
| 24442 | TRAFO_AXES_IN_6 | Axis assignment for the 6th transformation | F2 |
| 24444 | TRAFO_GEOAX_ASSIGN_TAB_6 | Geo-axis assignment for 6th transformation | F2 |
| 24446 | TRAFO_INCLUDES_TOOL_6 | Tool handling with active transformation 6 | F2 |
| 24450 | TRAFO_TYPE_7 | Definition of the 7th transformation in channel | F2 |
| 24452 | TRAFO_AXES_IN_7 | Axis assignment for the 7th transformation | F2 |
| 24454 | TRAFO_GEOAX_ASSIGN_TAB_7 | Geo-axis assignment for 7th transformation | F2 |
| 24456 | TRAFO_INCLUDES_TOOL_7 | Tool handling with active transformation 7 | F2 |
| 24460 | TRAFO_TYPE_8 | Definition of the 8th transformation in channel | F2 |
| 24462 | TRAFO_AXES_IN_8 | Axis assignment for the 8th transformation | F2 |
| 24464 | TRAFO_GEOAX_ASSIGN_TAB_8 | Geo-axis assignment for 8th transformation | F2 |
| 24466 | TRAFO_INCLUDES_TOOL_8 | Tool handling with active transformation 8 | F2 |
| 24470 | TRAFO_TYPE_9 | Definition of the 9th transformation in channel | F2 |
| 24472 | TRAFO_AXES_IN_9 | Axis assignment for the 9th transformation | F2 |
| 24474 | TRAFO_GEOAX_ASSIGN_TAB_9 | Geo-axis assignment for 9th transformation | F2 |
| 24476 | TRAFO_INCLUDES_TOOL_9 | Tool handling with active transformation 9 | F2 |
| 24480 | TRAFO_TYPE_10 | Definition of the 10th transformation in channel | F2 |
| 24482 | TRAFO_AXES_IN_10 | Axis assignment for the 10th transformation | F2 |
| 24484 | TRAFO_GEOAX_ASSIGN_TAB_10 | Geo-axis assignment for 10th transformation | F2 |
| 24486 | TRAFO_INCLUDES_TOOL_10 | Tool handling with active transformation 10 | F2 |
| 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) | |

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| 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 | |

## 7.2.3 Interrupts

The alarms which may occur in conjunction with the TRACYL transformation are explained in the Diagnostic Guide and in the online help in systems with MMC 101/102/103
**References:** /DA/, “Diagnostic Guide”

# 7.3 TRAANG (inclined axis)

## 7.3.1 Interface signals

| DB number | Bit, byte | Name | Refer-ence |
|---|---|---|---|
| **Channel-specific** | | | |
| 21, ... | 33.6 | Transformation active | F2 |

## 7.3.2 Machine data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Channel-specific ($MC_...)** | | | |
| 20110 | RESET_MODE_MASK | Definition of control basic setting after run-up and RESET/part program end | K2 |
| 20140 | TRAFO_RESET_VALUE | Basic transformation position | K2 |
| 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 | F2 |
| 24110 | TRAFO_AXES_IN_1 | Axis assignment for the 1st transformation | F2 |
| 24120 | TRAFO_GEOAX_ASSIGN_TAB_1 | Geo-axis assignment for 1st transformation | F2 |
| 24200 | TRAFO_TYPE_2 | Definition of the 2nd transformation in channel | F2 |
| 24210 | TRAFO_AXES_IN_2 | Axis assignment for the 2nd transformation | F2 |
| 24220 | TRAFO_GEOAX_ASSIGN_TAB_2 | Geo-axis assignment for 2nd transformation | F2 |
| 24300 | TRAFO_TYPE_3 | Definition of the 3rd transformation in channel | F2 |
| 24310 | TRAFO_AXES_IN_3 | Axis assignment for the 3rd transformation | F2 |
| 24320 | TRAFO_GEOAX_ASSIGN_TAB_3 | Geo-axis assignment for 3rd transformation | F2 |
| 24400 | TRAFO_TYPE_4 | Definition of the 4th transformation in channel | F2 |
| 24410 | TRAFO_AXES_IN_4 | Axis assignment for the 4th transformation | F2 |
| 24420 | TRAFO_GEOAX_ASSIGN_TAB_4 | Geo-axis assignment for 4th transformation | F2 |
| 24430 | TRAFO_TYPE_5 | Definition of the 5th transformation in channel | F2 |
| 24432 | TRAFO_AXES_IN_5 | Axis assignment for the 5th transformation | F2 |
| 24434 | TRAFO_GEOAX_ASSIGN_TAB_5 | Geo-axis assignment for 5th transformation | F2 |
| 24440 | TRAFO_TYPE_6 | Definition of the 6th transformation in channel | F2 |
| 24442 | TRAFO_AXES_IN_6 | Axis assignment for the 6th transformation | F2 |
| 24444 | TRAFO_GEOAX_ASSIGN_TAB_6 | Geo-axis assignment for 6th transformation | F2 |
| 24450 | TRAFO_TYPE_7 | Definition of the 7th transformation in channel | F2 |
| 24452 | TRAFO_AXES_IN_7 | Axis assignment for the 7th transformation | F2 |

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| 24454 | TRAFO_GEOAX_ASSIGN_TAB_7 | Geo-axis assignment for 7th transformation | F2 |
| 24460 | TRAFO_TYPE_8 | Definition of the 8th transformation in channel | F2 |
| 24462 | TRAFO_AXES_IN_8 | Axis assignment for the 8th transformation | F2 |
| 24464 | TRAFO_GEOAX_ASSIGN_TAB_8 | Geo-axis assignment for 8th transformation | F2 |
| 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 geo-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) | |
| 24760 | TRAANG_BASE_TOOL_2 | Distance of tool zero point from origin of geo-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) | |

## 7.3.3 Interrupts

The alarms which may occur in conjunction with the TRAANG transformation are explained in the Diagnostic Guide and in the online help in systems with MMC 101/102/103

**References:** /DA/, “Diagnostic Guide”

## 7.4 TRACON (chained transformations)

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channel-specific ($MC_...)** | | | |
| 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 | |

## 7.5 Non transformation-specific machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channel-specific ($MC_...)** | | | |
| 21110 | X_AXIS_IN_OLD_X_Z_PLANE | Coordinate system for automatic Frame definition | |
| 21090 | MAX_LEAD_ANGLE | Maximum permissible lead angle for orientation programming | |
| 21092 | MAX_TILT_ANGLE | Maximum permissible side angle for orientation programming | |
| 21100 | ORIENTATION_IS_EULER | Angle definition for orientation programming | F2 |

■

# SINUMERIK 840D/840Di/810D Description of Functions Extended Functions (FB2)

# Measurement (M5)

■

# 1 Brief Description

**Channel-specific measuring**

Channel-specific measuring is available with SW 1 and higher.
A trigger event which initiates the measuring operation and defines a corresponding measuring mode is programmed in a part program block. The instructions apply to all axes programmed in this particular block.

**Preset actual value memory and scratching**

**Preset actual value memory** is initiated by means of an MMC 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

- MMC operator action or a
- parts 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

- MMC operator action or via
- measuring cycles.

Communication with the NC takes place via predefined system variables.

**Workpiece/tool measurement**

The position of the workpiece in relation to an edge, a corner or a hole can be measured.

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 resultant 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.

**Axial measurement**

Axial measuring is available with SW 4.1 and higher.
A trigger event which initiates a measuring operation is programmed in a part program block. A measuring mode for the measurement is defined together with the axis in which the measurements must be taken.

**Measuring cycles**

A description of how to handle measuring cycles can be found in

**References:** /FB III/, Measuring Cycles (M4)

■

# Notes

# Detailed Description 2

## 2.1 Hardware requirements

### 2.1.1 Suitable probes

**General information**

In order to sense the dimensions of tool and workpiece, a touch trigger probe that outputs a constant signal (not a pulse) on deflection is required.

The sensor must operate virtually bounce-free. Most sensors can be adjusted mechanically to ensure that they operate in this manner.

Different types of probe supplied by a variety of manufacturers are available on the market. Probes are therefore divided into three groups according to the number of directions in which they can be deflected (see figure below).

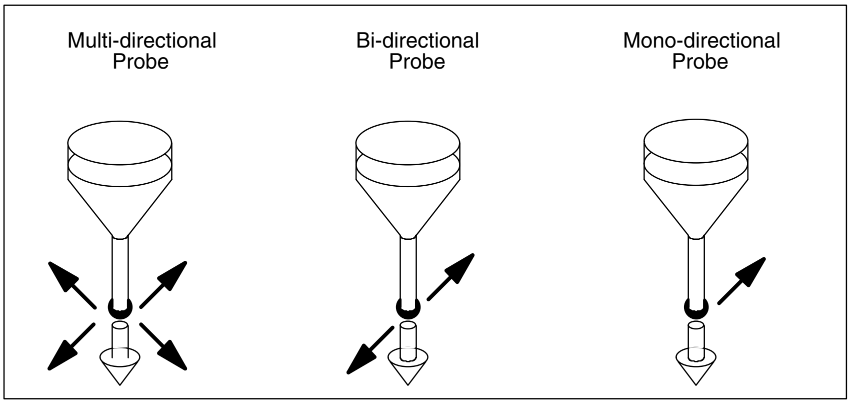



Fig. 2-1 Probe types

Table 2-1 Assignment between probe type and application

| Probe type | Turning machines | | Milling and machining centers |
|---|---|---|---|
| | Tool measurements | Workpiece measurements | Workpiece measurements |
| Multi-directional | X | X | X |
| Bi-directional | _ | X | X |
| Mono-directional | _ | _ | X |

Bi-directional probes must be used on turning machines for workpiece measurements, whereas a mono-probe can also be used for this purpose on milling and machining centers.

**Multidirectional probe (3D)**

This probe type can be used unconditionally for measuring tool and workpiece dimensions.

**Bidirectional probe**

This probe type is applied in the same way as a mono-probe in milling and machining centers. Bi-directional probes can be used to take workpiece measurements on turning machines.

**Monodirectional probe**

This probe type can be used subject to some restrictions to take workpiece measurements on milling and machining centers.

**Spindle position with mono probe**

To be able to use this probe type on milling and machining centers, it must be possible to position the spindle with NC function SPOS and to transfer the switching signal from the probe over 360° to the receiver station (on machine stator).

The probe must be mechanically aligned in the spindle such that it can take measurements in the following directions when the spindle is positioned at 0 degrees.

Table 2-2 Spindle positions for alignment of probe

| | **Measurements at 0 degrees spindle position** |
|---|---|
| X–Y plane G17 | Positive X direction |
| Z–X plane G18 | Positive Z direction |
| Y–Z plane G19 | Positive Y direction |

The measurement takes longer with a mono-probe as the spindle needs to be positioned several times with SPOS in the measuring cycle.

## 2.1.2 Probe connection

**Connection to 840D**

The probe is connected to the SINUMERIK 840D or 810D system via the I/O device interface X121 located on the front plate of the NCU module.

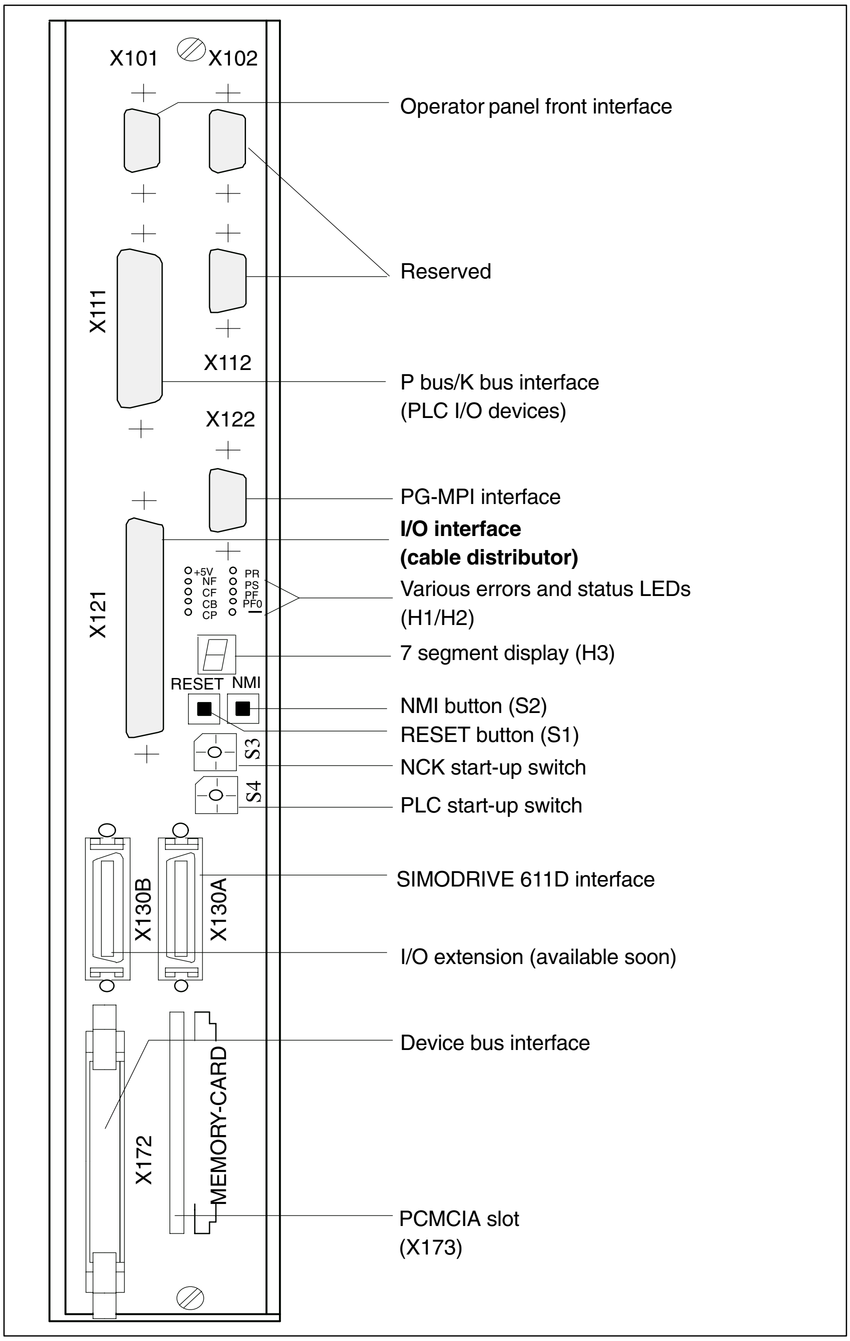



Fig. 2-2 Interfaces, control and display elements of NCU module

**Connection to 840Di**

The probe is connected to the SINUMERIK 840Di via the I/O interface X121 of the MCI board extension module (option).

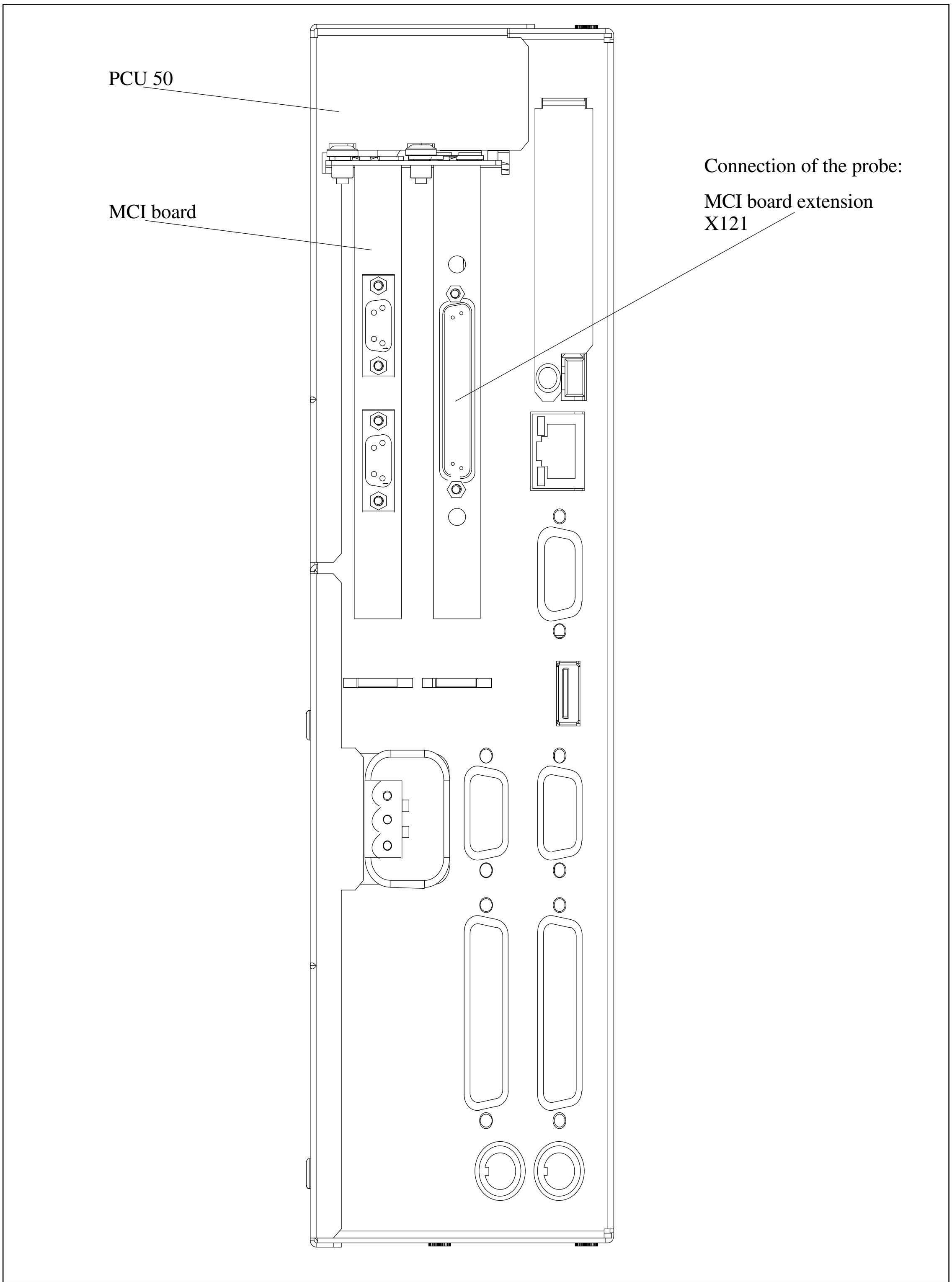



Fig. 2-3 Interfaces of the SINUMERIK 840Di (PCU 50, MCI board and MCI board extension)

**Connection to 810D**

The probe is connected to the SINUMERIK 810D via the I/O device interface X121 located on the front plate of the NCU module.

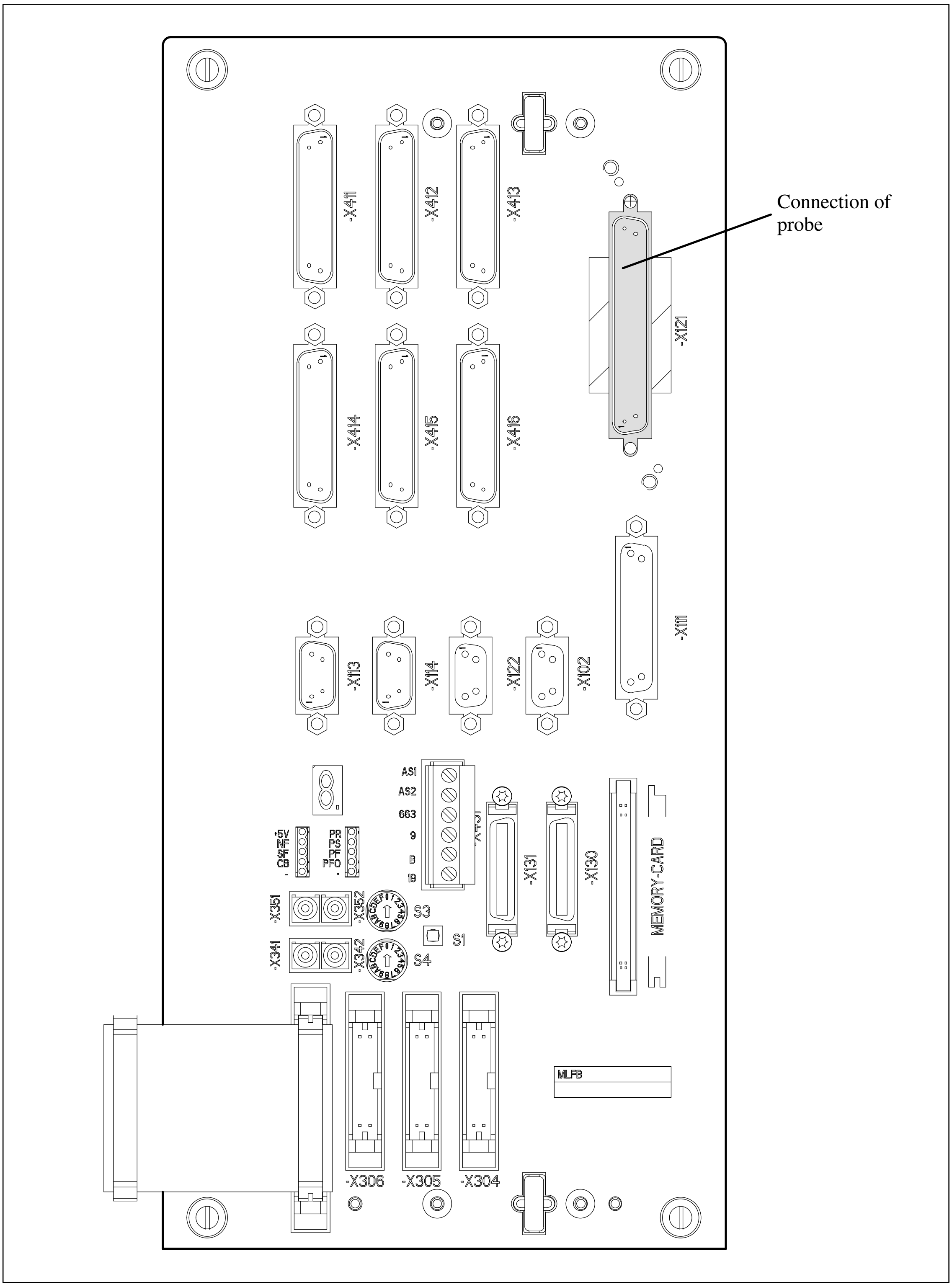



Fig. 2-4 Interfaces, operating and display elements on SINUMERIK 810D

**Interface**

The interface connection for a probe is made via the

- **I/O interface**

  37-pin D-sub plug connector (X121), a **maximum of 2** probes can be connected;

The 24 V load power supply is also connected on this connector.

Table 2-3 Extract from PIN assignment table for X121 front connectors

| | PIN | | Designation |
|---|---|---|---|
| X121 | | | External power supply |
| | 1 | M24EXT | External ground |
| | 2 | M24EXT | External ground |
| | ... | ... | ... |
| | | | Connection probe 1 |
| | 9 | MEPUS 0 | Measuring pulse signal input |
| | 10 | MEPUC 0 | Measuring pulse common input |
| | ... | ... | ... |
| | | | External power supply |
| | 20 | P24EXT | P 24 V external |
| | 21 | P24EXT | P 24 V external |
| | ... | ... | ... |
| | | | Connection probe 2 |
| | 28 | MEPUS 1 | Measuring pulse signal input |
| | 29 | MEPUC 1 | Measuring pulse common input |
| | ... | ... | ... |

The interfaces and pin assignments are illustrated and described in detail in:

**References:** /PHD/, Hardware Configuring Guide

**PROFIBUS-DP drives**

It is possible to operate a distributed probe direct on the PROFIBUS DP drive for SINUMERK 840D with an NCU 573.2/3/4. This method is more accurate than NC interpolation of cyclic position values from a centralized probe.

The type of measuring function for PROFIBUS DP drives, e.g. with SIMODRIVE 611 universal, is specified as follows with MD 13210: MEAS_TYPE:

Value = 0: Centralized probe connected to the NC or
Value = 1: Distributed probe for **all** drives.

SIMODRIVE 611 universal drives support the measurement functionality of distributed probes by storing the actual encoder value in the hardware concurrent to the measurement signal edge. The more accurate measurement method of a distributed probe is preferred for PROFIBUS DP drives.

SIMODRIVE 611 digital drives continue to be operated with a centralized probe at connector X121 on the SINUMERK 840D/840Di/810D.

**References:** /BHA/, Absolute Value Encoders with PROFIBUS DP
/FBU/, SIMODRIVE 611 universal

## 2.2 Channel-specific measuring

### 2.2.1 Software requirements

**NC software version**

Channel-specific measuring functionality is available with the current software versions.

**Operator panel front software versions**

The “Measurement result display” and “Parameter assignment” functions are available via input dialog with

- MMC software version
- HMI Advanced and HMI Embedded

### 2.2.2 Measuring mode

**Measurement commands MEAS and MEAW**

The measuring operation is activated from the parts program. A trigger event and a measuring mode are programmed.
Two different measuring modes are available:

- 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 GEO axis is programmed in a measuring block, then the measured values are stored for all current GEO axes.

If an axis which is taking part in a transformation is programmed in a measuring block, then the measured values of all of the axes taking part in this transformation are stored.

---

**Measuring probe status**

It is possible to scan the probe status directly in the part program and in synchronized actions.

$A_PROBE[n] with n=probe
$A_PROBE[n]==1: probe deflected
$A_PROBE[n]==0: probe not deflected

## 2.2.3 Measurement results

**Read measurement results in PP**

The results of the measurement commands are stored in system data of the NCK and can be read via system variables in the parts program.

- **System variable $AC_MEA[No]**
  Scan status signal of measurement job.
  [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 parts program.
- **System variable $AA_MM[axis]**
  Access to measurement result in machine coordinate system.
  Read in parts program and in synchronized actions.
  [Axis] stands for the name of the measurement axis (X, Y, ...).
- **System variable $AA_MW[axis]**
  Access to measurement result in workpiece coordinate system.
  Read in parts program and in synchronized actions.
  [Axis] stands for the name of the measurement axis (X, Y, ...).

**References:** /PGC/, Programming Guide Cycles

**PLC service display**

The functional test for the probe is conducted via an NC program.

The measuring signal can be checked at the end of the program in the diagnostic menu “PLC status”.

Table 2-4 Status display for measuring signal

| | **Status display** |
|---|---|
| Probe 1 deflected | DB10 DB B107.0 |
| Probe 2 deflected | DB10 DB B107.1 |

The current measuring status of the axis is displayed by means of the interface signal DB(31–48) DBX62.3.

Bit 3=1: Measurement active
Bit 3=0: Measurement not active

---

**Note**

This signal can be displayed for all measuring functions and also read in synchronized actions with $AA_MEAACT[axis].
**References:** /FB2/, S5, Synchronized Actions

---

## 2.3 Zero setting, workpiece and tool measuring

### 2.3.1 PRESET and scratching

**Preset actual value memory**

**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
- parts 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 measurement **and** the tool measurement.

The measurements can be initiated via

- HMI operator action or via
- measuring cycles.

Communication with the NC takes place via predefined system variables.

**References:** /FB1/, K2 “Axes, Coordinate Systems, Frames”
/PGA/, Tables “List of System Variables”

### 2.3.2 Workpiece measuring

**Workpiece measuring**

The position of the workpiece in relation to an edge, a corner or a hole can be measured.

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 resultant offsets can be entered in a selected frame.

**Variable interface**

The variable interface comprises several system variables which are categorized as either

- Input values or
- Output values

## Input values

Input values must be written by the HMI or the cycles. The output values are the results of the calculations.

Table 2-5 All validity bits for the measurement types of variable $AC_MEAS_VALID

| Bit | $AC_MEAS_VALID Input variable | 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 edge, corner, hole |
| 5 | $AC_MEAS_WP_SETANGLE | Setpoint workpiece position angle $\alpha$; $-90 < \Phi < 180$ |
| 6 | $AC_MEAS_CORNER_SETANGLE | Setpoint angle of intersection $\Phi$ of corner $0 < \Phi < 180$ |
| 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 to 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 |

**Specifications for current measurement**

Each input variable always sets the corresponding bit in system variable $AC_MEAS_VALID when writing to the interface.

**All** input variables of $AC_MEAS_VALID should be declared as invalid before **every** measurement. If the validity bits are not reset, the input values also remain valid for the next calculation.

Table 2-6 Inputs values for the calculation and the measuring points

| Type | System variable | Meaning |
|---|---|---|
| INT | $AC_MEAS_VALID | Validity bits for the calculation |
| 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 |
| INT | $AC_MEAS_P1_COORD* | Coordinate system of 1st measuring point |
| INT | $AC_MEAS_P2_COORD* | Coordinate system of 2nd measuring point |
| INT | $AC_MEAS_P3_COORD* | Coordinate system of 3rd measuring point |
| INT | $AC_MEAS_P4_COORD* | Coordinate system of 4th measuring point |
| INT | $AC_MEAS_SET_COORD* | Coordinate system of setpoint |
| INT | $AC_MEAS_LATCH4[0..3] | Write measuring points 1 to 4 for all axes with the current WCS axis actual values |
| INT | $AA_MEAS_P1_VALID[axis] | Write individual axis actual values from P1 |
| INT | $AA_MEAS_P2_VALID[axis] | Write individual axis actual values from P2 |
| INT | $AA_MEAS_P3_VALID[axis] | Write individual axis actual values from P3 |
| INT | $AA_MEAS_P4_VALID[axis] | Write individual axis actual values from P4 |

* Coordinate system in which point was measured
0: WCS is default setting
1: BCS
2: MCS

**Note**

Set all validity bits (input values) to invalid:
$AC_MEAS_VALID = 0

All axis actual values of the appropriate measuring point are invalidated by:
$AC_MEAS_LATCH = 0

## Measuring points

Variables $AC_MEAS_POINT[1..4] are used to specify the measuring points. Each individual measuring point can be written or picked up.

## Actual values

Variable $AC_MEAS_LATCH can only be written. With an assignment of $AC_MEAS_LATCH = 1

All axis actual values are picked up at the appropriate measuring point. The index of $AC_MEAS_LATCH varies between 0 and 3 depending on measuring points 1 to 4.

Axis actual values of the x axis in the 1st measuring point are, e.g.

$AA_MEAS_P1_VALID[x] = 0 Axis actual value is invalid
$AA_MEAS_P1_VALID[x] = 1 Axis actual value is picked up

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 such that the measurement complies with the setpoints specified by the user.

Table 2-7 Inputs values for the user setpoints

| Type | System variable | Meaning |
|---|---|---|
| REAL | $AA_MEAS_SETPOINT[axis] | Setpoint position of edge, corner, hole |
| INT | $AA_MEAS_SP_VALID[axis] | 1: Setpoint position of axis is valid / 0: Invalid |
| REAL | $AC_MEAS_WP_SETANGLE | Setpoint workpiece position angle<br>$\alpha$; –90 < $\Phi$ < 180 |
| REAL | $AC_MEAS_CORNER_SETANGLE | Setpoint angle of intersection $\Phi$ of corner<br>0 < $\Phi$ < 180 |

The following measuring points are irrelevant and not evaluated:

When setpoint workpiece pos. angle $\alpha$ of the 2nd measuring point is entered.
When setpoint angle of intersection $\Phi$ of the 4th measuring point is entered.

**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.

INT $AC_MEAS_T_NUMBER Selected tool
INT $AC_MEAS_D_NUMBER Selected cutting edge

**Measurements with 3D probe**

In the case of measurements with the 3D probe, the radius of the tool is already compensated with reference to the measuring point, and so 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:

INT $AC_MEAS_TOOL_MASK Tool settings bit mask

0x0: All tool lengths are considered (default setting).
0x1: Radius of the tool 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: Length of the tool is not included in the calculation

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 input explicitly, it is determined by the selected plane. With G17 the approach direction is in the z direction,
with G18 in the y direction and with G19 in the x direction.

**Approach direction**

The direction of approach is required only for edge, groove, web and tool measurements:

INT $AC_MEAS_DIR_APPROACH =

0: +x
1: –x
2: +y
3: –y
4: +z
5: –z

**Plane setting**

The plane must be specified for the calculation.

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 workpieces are measured, translational offsets can be entered in the fine offset component of the selected frame. Variable $AC_MEAS_FINE_TRANS is used for this purpose.

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 applies when $AC_MEAS_FINE_TRANS = 1:
The compensation value is entered in the fine component of the translation and transformed according to the selected frame.
The coarse offset component remains unchanged.

When $AC_MEAS_FINE_TRANS = 0 or nothing has been written, the following applies:
The compensation value is entered in the coarse offset and transformed accordingly.

If MD 18600: MM_FRAME_FINE_TRANS is not preset to 1:
The compensation value is always entered in the coarse offset.

**Calculated frame**

When a workpiece is measured, the calculated frame is entered in the specified frame.

INT $AC_MEAS_FRAME_SELECT =

| | | |
|---|---|---|
| | Active frames: | |
| 0: | $P_SETFR | System frame |
| 10..25: | $P_CHBFR[0..15] | Channel-specific basic frame |
| 50..65: | $P_NCBFR[0..15] | NCU-global basic frames |
| 100..199: | $P_UIFR[0..99] | Settable frames |
| 1010..1025: | $P_CHBFR[0..15] | Channel-spec. basic frame with act. G500 |
| 1050..1065: | $P_NCBFR[0..15] | NCU-global basic frame with active G500 |
| | Frames in the data management: | |
| 2000: | $P_SETFR | System frame |
| 2010..2025: | $P_CHBFR[0..15] | Channel-specific basic frame |
| 2050..2065: | $P_NCBFR[0..15] | NCU-global basic frames |
| 2100..2199: | $P_UIFR[0..99] | Settable frames |
| 3010..3025: | $P_CHBFR[0..15] | Channel-spec. basic frame with act. G500 |
| 3050.0.3065: | $P_NCBFR[0..15] | NCU-global basic frame with active G500 |

The MEASURE( ) function calculates frame $AC_MEAS_FRAME according to the specified frame.

In the case of values
from **0 to 1065**, the calculation is performed with the aid of the active frame.

from **2000 to 3065**, the calculation is performed with reference to the selected frame in the data management. The selection of a frame in the 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 the data management. In this case, the calculation is performed as if the frame were active in the chain.
The measuring point is transformed to the selected system and the selected frame is determined with the aid of the overall frame including the selected frame. Preset actual value memory is active only after compensation and activation of the frame.

With **G500** active (1010..1025, 1050..1065, 3010..3025, 3050..3065), the target frame is calculated such that G500 must be active after the frame is selected so that the setpoint position can be calculated.

**Conversion to another coordinate system**

If a position is to be converted to a position of another coordinate system, the following variables can be used to specify the composition of the desired frame chain:

| | |
|---|---|
| INT $AC_MEAS_CHSFR<br>bit mask MD 28082: | System frame according to<br>MM_SYSTEM_FRAME_MASK |
| INT $AC_MEAS_NCBFR<br>bit mask MD 18602: | Global basic frames according to<br>MM_NUM_GLOBAL_BASE_FRAME |
| INT $AC_MEAS_CHBFR<br>bit mask MD 28081: | Channel basic frame according to<br>MM_NUM_BASE_FRAMES |
| INT $AC_MEAS_UIFR<br>value 0 .. 99 in MD 18601: | Settable frame according to<br>MM_NUM_GLOBAL_USER_FRAME |
| INT $AC_MEAS_PFRAME<br>0: Programmable frame is<br>1: Programmable frame is not | Programmable frame according to<br>included<br>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 applied in other input parameters used in the different measurement types

REAL $AC_MEAS_INPUT[n] n =0..9 measuring input parameter

The control action of the measuring input parameters is described with the measuring methods.

**Selection of measurement**

The measurement is selected by means of the following variables:

| | |
|---|---|
| INT $AC_MEAS_TYPE | ; Select measurement type |
| 0: Default | ; Measurement: |
| 1: Edge_x | ; of an edge x |
| 2: Edge_y | ; of an edge y |
| 3: Edge_z | ; of an edge z |
| 4: Corner_1 | ; of a corner 1 |
| 5: Corner_2 | ; of a corner 2 |
| 6: Corner_3 | ; of a corner 3 |
| 7: Corner_4 | ; of a corner 4 |
| 8: Hole | ; of a hole |
| 9: Stud | ; of a shaft |
| 10: ToolLength | ; of tools (tool length) |
| 11: ToolDiameter | ; of tools (tool diameter) |
| 12: Slot | ; of a groove |
| 13: Plate | ; of a web |
| 14: Set_Pos | ; Preset actual value memory for geo and<br>; special axes |
| 15: Set_AuxPos | ; Preset actual value memory for special<br>; axes only |
| 16: Edge_2P | ; Measurement of oblique edges |
| 17: Plane_Angles | ; Angle of a plane. |
| 18: Plane_Normal | ; Angle of a plane with specified setpoint. |
| 19: Dimension_1 | ; 1-dimensional setpoint input. |
| 20: Dimension_2 | ; 2-dimensional setpoint input. |
| 21: Dimension_3 | ; 3-dimensional setpoint input. |

The following variables are added to the measurement selection:

| | |
|---|---|
| INT $AC_MEAS_TYPE | ; Select measurement type |
| | ; ShopTurn: Measuring: |
| 22: ToolMagnifier | ; Tool lengths with zoom-in function |
| 23: ToolMarkedPos | ; Tool lengths with stored position |
| 24: Coordinate transformation | ; Transformation of a position to a posi-<br>; tion of another coordinate system |
| 25: Rectangle | ; Measuring a rectangle |
| 26: Save | : Saving data management frames |
| 27: Restore | : Restoring data frames |

The individual methods are listed under “Types of workpiece measurement” and explained in more detail by means of an appropriate programming example.

## 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 parts program. The result frame is calculated in accordance with the selected frame.

If no frame has been selected, the result frame determines the final translation and rotation in the WCS. PI service _N_SETUDT /R7/ and

parameter type no. 7 can be used to enter this frame in the selected frame. Once it has been entered, the result frame is deleted.

Table 2-8 Output values of calculation results

| Type | System variable | Meaning |
|---|---|---|
| FRAME | $AC_MEAS_FRAME | Result frame |
| REAL | $AC_MEAS_WP_ANGLE | Calculated workpiece position angle $\alpha$ |
| REAL | $AC_MEAS_CORNER_ANGLE | Calculated angle of intersection $\Phi$ |
| REAL | $AC_MEAS_DIAMETER | Calculated diameter |
| REAL | $AC_MEAS_TOOL_LENGTH | Calculated tool length |
| REAL | $AC_MEAS_RESULTS[10] | Calculation results (type-dependent) |

## Calculation method

**Activation**

The calculation is activated by an HMI operator action with PI service _N_SETUDT /R7/. This PI service receives a

parameter type no. 1 – active tool offset
no. 2 – active basic frames
no. 3 – active settable frame
no. 4 – global basic frames
no. 5 – global settable frames
no. 6 – workpiece zero or tool length calculation
no. 7 – activate workpiece zero (write scratching).

PI service _N_SETUDT /R7/ has been expanded as follows:

no. 8 – activate external zero offset
no. 9 – activate active tool carrier, TCOABS and PAROT

The modification becomes apparent immediately in the reset state; in the stop state, the frame is not applied until 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 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. When execution of the part program is resumed, the distance to go in the interrupted block is deleted and the axis approaches the end point of the next block from its current position.

It is therefore possible to operate a spindle in MDA mode or execute preset actual value memory and scratching or another measurement, e.g. with M0, in the part program in the Stop state.

**Measuring cycles**

The calculation in the measuring cycles is performed according to the predefined function:

INT MEASURE( )

MEASURE() produces a result frame which can be read via $AC_MEAS_FRAME.

The result is the translation and rotation from the

- setpoints, converted to the selected frame.

The result frame is calculated as follows:

- The concatenated total frame equals the concatenation of the total frame (prior to measurement) with the calculated translation and rotation.

---

**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.

It must be noted that the result frame may need to be deleted beforehand.

---

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.

**Semaphore variable**

The following variable serves to protect against mutual overwriting

INT $AC_MEAS_SEMA

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.

The HMI does not use the interface if the variable is set to 1.

**Error messages**

If the wrong procedure is adopted, up to 18 different return values can be output via the following predefined functions:

Table 2-9 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 $\alpha$ is wrong |
| 11 | MEAS_WRONG_PHI | Phi $\Phi$ 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 fault |

**Logon error**

If the user does not log on, group error number 0xD003 is always generated. If DIAGN:errCodeSeNrGent is used to log on, then several P_SETUDT error codes are available.

**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 2-10 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 |

## 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 calc. 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 input variable

| | |
|---|---|
| INT $AC_MEAS_SCALEUNIT | Unit of measurement for input and outputvariables |
| 0: Unit of measurement with reference to | active G code G70/G700 is INCH active G code G71/G701 is METRIC: |
| 1: Unit of measurement is determined | by configuration, the system of (default setting) measurement can be set via OPI |

If the variable is not written, the value 0 is always the default setting. Examples for **basic system metric:**

```
G70                                   ; basic system metric
$AC_MEAS_POINT1[x] = $AA_IW[x]        ; $AA_IW[x] returns
$AC_MEAS_POINT1[x] = 10               ; 10 mm
G71                                   ; basic system metric
$AC_MEAS_POINT1[x] = $AA_IW[x]        ; $AA_IW[x] returns
$AC_MEAS_POINT1[x] = 10               ; 10 mm
G700                                  ; Inch value
$AC_MEAS_POINT1[x] = $AA_IW[x]        ; $AA_IW[x] returns
$AC_MEAS_POINT1[x] = 10               ; 10 Inch
G710                                  ; Metric value
$AC_MEAS_POINT1[x] = $AA_IW[x]        ; $AA_IW[x] returns
$AC_MEAS_POINT1[x] = 10               ; 10 mm
```

**DIAMON or DIAMOF**

Diameter programming is set via machine data:

```
MD 20100: DIAMETER_AX_DEF = "X"           ; Transverse axis is x
MD 20150: GCODE_RESET_VALUES[28] = 2      ; DIAMON
MD 20360:TOOL_PARAMETER_DEF_MASK =        ; TLC, frames and
                          'B1001010'      ; actual values in diameter
```

Axis positions in the MCS are not computed as a diameter value.

| Diameter programming | Meaning |
|---|---|
| DIAMON or DIMOF active | The calculated tool lengths and frame components do not depend on the active G code. |
| DIAMON active | The measured positions and setpoint positions are read and written depending on DIAMON. |
| DIAMON active | The translations in the frames are calculated as a diameter in the transverse axis. |
| **Value precision:** Values are rounded to six decimal places | Position values in mm, inches or degrees are calculated and displayed accurate to six decimal places. |

## Types of workpiece measurement

### 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.

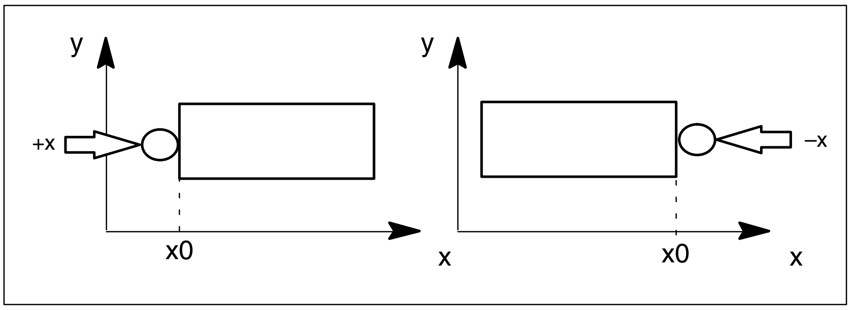



Fig. 2-5 x edge

Table 2-11 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 | 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 | 1 |

* optional

Table 2-12 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 measured edge |

### x edge measurement

### Programming example

```
DEF INT RETVAL                        ;
DEF FRAME TMP                         ;

$TC_DP1[1,1] = 120                    ; Typ
$TC_DP2[1,1] = 20                     ; 0
$TC_DP3[1,1] = 10                     ; (z) length offset 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                                   ;

$AC_MEAS_VALID = 0                    ; Invalidate all input values

g1 x–1 y–3                            ; Approach 1st 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

                                      ; Set setpoint position of edge
$AA_MEAS_SETPOINT[x] = 0              ;
$AA_MEAS_SETPOINT[y] = 0              ;
$AA_MEAS_SETPOINT[z] = 0              ;

$AC_MEAS_ACT_PLANE = 0                ; G17 is the measurement plane

$AC_MEAS_FRAME_SELECT = 101           ; Select frame (G54)

                                      ; Select tool
$AC_MEAS_T_NUMBER = 1                 ;
$AC_MEAS_D_NUMBER = 1                 ;

$AC_MEAS_TYPE = 1                     ; Set measurement type x edge

                                      ; Perform calculation
RETVAL = MEASURE()                    ;

if RETVAL <> 0                        ;
setal(61000 + RETVAL)                 ;
endif

$P_IFRAME = $AC_MEAS_FRAME            ;

$P_UIFR[1] = $P_IFRAME                ; Write system frame in data management

g1 x0 y0                              ; Approach the edge

m30
```

**Measurement of an y edge ($AC MEAS TYPE = 2)**

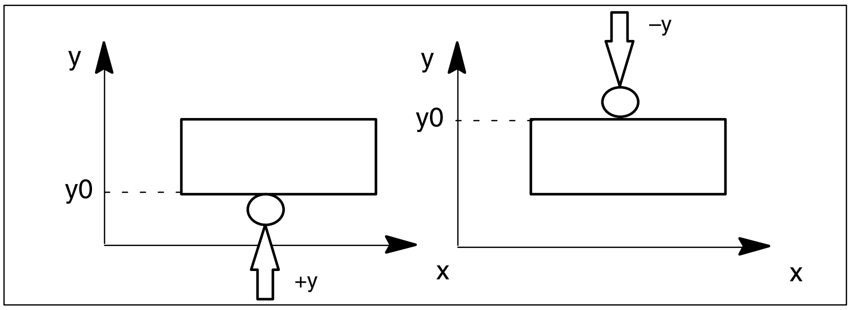



Fig. 2-6 y edge

Table 2-13 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 | 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 | 2 |

* optional

Table 2-14 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 measured edge |

## Measurement of an z edge ($AC MEAS TYPE = 3)

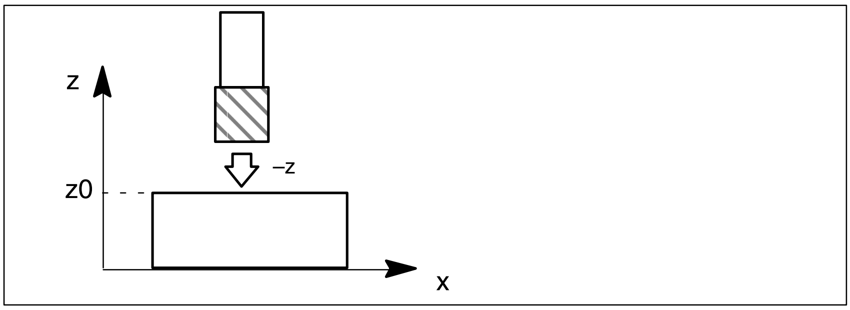



Fig. 2-7 z edge

Table 2-15 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: +y, 5: –y |
| $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 | 3 |

* optional

Table 2-16 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 measured edge |

**Measurement of an**

**Corner C1 –C4 ($AC MEAS TYPE = 4, 5, 6, 7)**

A corner is uniquely defined through the approach of four measuring points P1 to P4. When the angle of intersection $\Phi$ 3 measuring points can be sufficient. If the angle of intersection $\Phi$ and the workpiece position angle $\alpha$ are known, two measuring points, P1 and P3, are sufficient.

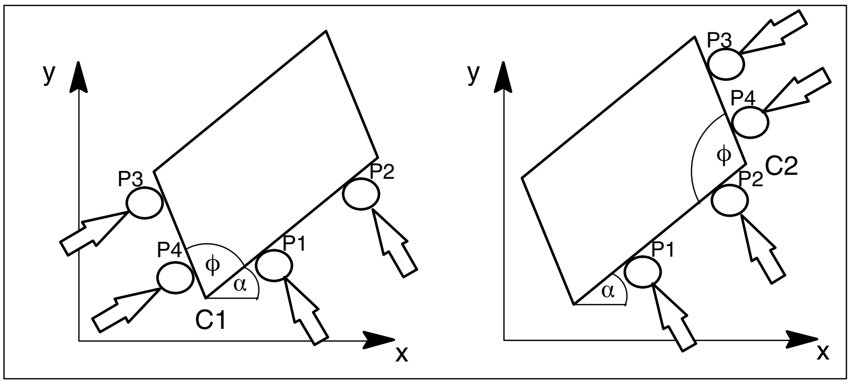



Fig. 2-8 Corner C1 Corner C2

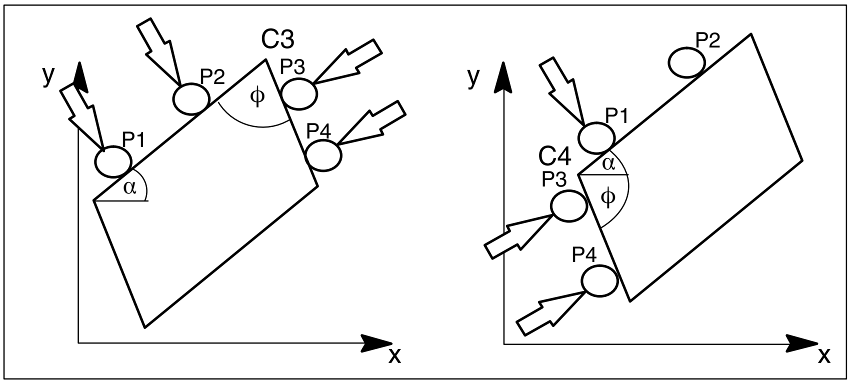



Fig. 2-9 Corner C3 Corner C4

Table 2-17 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 with $AC_MEAS_WP_SETANGLE |
| $AA_MEAS_POINT3[axis] | Measuring point 3 |
| $AA_MEAS_POINT4[axis] | Measuring point 4 irrelevant with $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*<br>=0: Measurement for outer corner<br>=1: Measurement for inner corner |
| $AC_MEAS_TYPE | 4, 5, 6, 7 |

* optional

Table 2-18 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 |

**Corner measurement C1**

### Programming example

Corner with three measuring points P1, P3 and P4 and known angle of intersection $\Phi$ degrees and unknown workpiece position angle $\alpha$.

```
DEF INT RETVAL                    ;
DEF FRAME TMP                     ;

$TC_DP1[1,1]=120                  ; Typ
$TC_DP2[1,1]=20                   ; 0
$TC_DP3[1,1]=10                   ; (z) length offset 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                                ;

$P_CHBFRAME[0] = crot(z,45)        ;
$P_IFRAME[x,tr] = –sin(45)         ;
$P_IFRAME[y,tr] = –sin(45)         ;
$P_PFRAME[z,tr] = –45              ;

$AC_MEAS_VALID = 0                 ; Invalidate all input values

g1 x–1 y–3                         ; Approach 1st measuring point
$AC_MEAS_LATCH[0] = 1              ; Pick up measuring point P1

g1 x–4 y4                          ; Approach 3rd measuring point
$AC_MEAS_LATCH[2] = 1              ; Pick up measuring point P3

g1 x–4 y1                          ; Approach 4th measuring point
$AC_MEAS_LATCH[3] = 1              ; Pick up measuring point P4

                                   ; Set position setpoint to (0, 0, 0)
$AA_MEAS_SETPOINT[x] = 0           ;
$AA_MEAS_SETPOINT[y] = 0           ;
$AA_MEAS_SETPOINT[z] = 0           ;

$AC_MEAS_CORNER_SETANGLE = 90; Input setpoint angle of intersection Φ
$AC_MEAS_ACT_PLANE = 0             ; Plane for measurement is G17
$AC_MEAS_FRAME_SELECT = 0          ; Select frame – SETFRAME

                                   ; Select tool
$AC_MEAS_T_NUMBER = 1              ;
$AC_MEAS_D_NUMBER = 1              ;

$AC_MEAS_TYPE = 4                  ; Set measurement type to corner 1

                                   ; Perform calculation
RETVAL = MEASURE()                 ;

if RETVAL <> 0
setal(61000 + RETVAL)
endif

if $AC_MEAS_CORNER_ANGLE <> 90 ; Scan known cutting angle Φ setpoint
setal(61000 + $AC_MEAS_CORNER_ANGLE)
endif

$P_SETFRAME = $AC_MEAS_FRAME

$P_SETFR = $P_SETFRAME             ; Write system frame in data management

g1 x0 y0                           ; Approach the corner

g1 x10                             ; Machine rectangle
y10                                ;
x0                                 ;
y0                                 ;

m30
```

**Measurement of an 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 4 points are specified, the circle is adjusted according to the smallest square error method. The circle is calculated such that the sum of the square distances of the points on the resulting circle is minimal. The adjustment quality can be viewed.

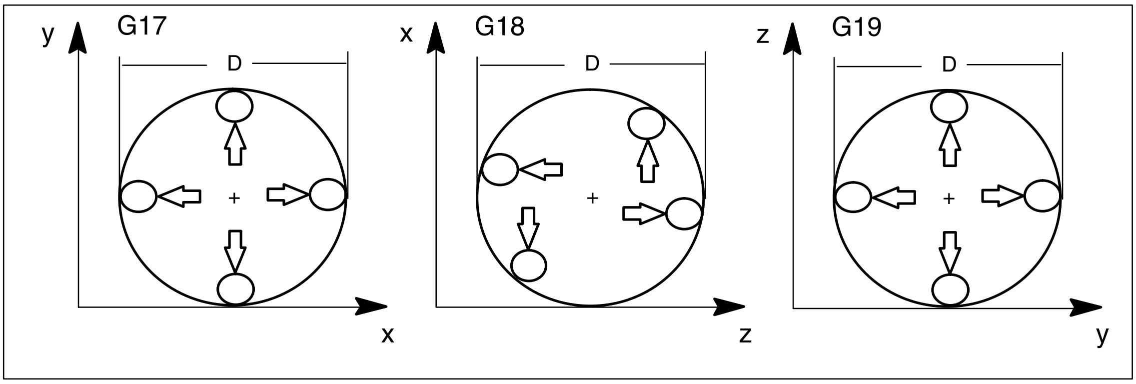



Fig. 2-10 Hole

Table 2-19 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 programmed, center is determined by 4 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

Table 2-20 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 calculated center point |
| $AC_MEAS_RESULTS[1] | Ordinate of calculated center point |
| $AC_MEAS_RESULTS[2] | Applicate of calculated center point |
| $AC_MEAS_RESULTS[3] | Quality scale for circle adjustment: Sum of square distances |

**Measure hole**

## Programming example

```
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                                  ;

$AC_MEAS_VALID = 0                   ; Invalidate all input values

g1 x–3 y0                            ; Approach 1st 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                             ; Approach 2nd 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                             ; Approach 3rd measuring point
                                     ;
$AA_MEAS_POINT3[x] = $AA_IW[x]       ;
$AA_MEAS_POINT3[y] = $AA_IW[y]       ;
$AA_MEAS_POINT3[z] = $AA_IW[z]       ;

                                     ; Set setpoint position of center
$AA_MEAS_SETPOINT[x] = 0             ;
$AA_MEAS_SETPOINT[y] = 0             ;
$AA_MEAS_SETPOINT[z] = 0             ;

$AC_MEAS_ACT_PLANE = 0               ; G17 is the measurement plane
$AC_MEAS_FRAME_SELECT = 0            ; Select frame – SETFRAME

                                     ; Select tool
$AC_MEAS_T_NUMBER = 1                ;
$AC_MEAS_D_NUMBER = 1                ;

$AC_MEAS_TYPE = 8                    ; Set measurement type to hole
```

```
                                      ; Perform calculation
RETVAL = MEASURE()                    ;

if RETVAL <> 0
setal(61000 + RETVAL)
endif

if $AC_MEAS_DIAMETER <> 10            ; Scan known diameter
setal(61000 + $AC_MEAS_WP_ANGLE)
endif

$P_SETFRAME = $AC_MEAS_FRAME

$P_SETFR = $P_SETFRAME                ; Write system frame in data management

g1 x–3 y0                             ; Approach P1

g2 I = $AC_MEAS_DIAMETER / 2          ; Machine hole in relation to arc center

m30
```

**Measurement of an**

**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 4 points are specified, the circle is adjusted according to the smallest square error method. The circle is calculated such that the sum of the square distances of the points on the resulting circle is minimal. The adjustment quality can be viewed.

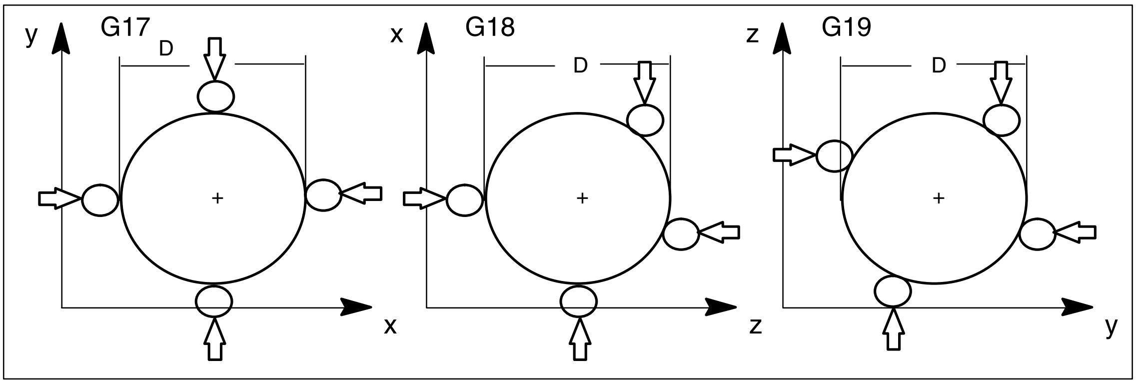



Fig. 2-11 Shaft

Table 2-21 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 |
| $AA_MEAS_POINT4[axis] | When programmed, center is determined by 4 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

Table 2-22 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 calculated center point |
| $AC_MEAS_RESULTS[1] | Ordinate of calculated center point |
| $AC_MEAS_RESULTS[2] | Applicate of calculated center point |
| $AC_MEAS_RESULTS[3] | Quality scale for circle adjustment: Sum of square distances |

**Measurement of an 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.

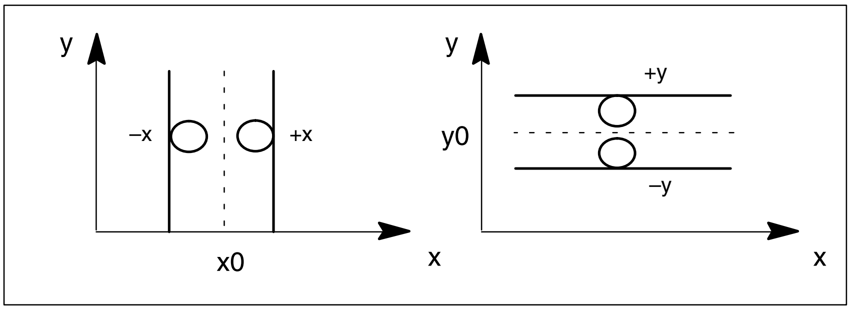



Fig. 2-12 Groove

Table 2-23 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 1st point. *<br>0: +x, 1: –x, 2: +y, 3: –y, 4: +z, 5: –z |
| $AC_MEAS_TYPE | 12 |

* optional

Table 2-24 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 |

**Groove measurement**

## Programming example

Groove measurement with approach direction in x

```
DEF INT RETVAL                      ;
DEF FRAME TMP                       ;

$TC_DP1[1,1] = 120                  ; Typ
$TC_DP2[1,1] = 20                   ; 0
$TC_DP3[1,1] = 10                   ; (z) length offset 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                                 ;

$AC_MEAS_VALID = 0                  ; Invalidate all input values

g1 x–2                              ; Approach 1st 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                               ; Approach 2nd measuring point
                                    ;
$AA_MEAS_POINT2[x] = $AA_IW[x]      ;
$AA_MEAS_POINT2[y] = $AA_IW[y]      ;
$AA_MEAS_POINT2[z] = $AA_IW[z]      ;

                                    ; Set setpoint position of groove center
$AA_MEAS_SETPOINT[x] = 0            ;
$AA_MEAS_SETPOINT[y] = 0            ;
$AA_MEAS_SETPOINT[z] = 0            ;

$AC_MEAS_DIR_APPROACH = 0           ; Set approach direction +x
$AC_MEAS_ACT_PLANE = 0              ; Plane for measurement is G17
$AC_MEAS_FRAME_SELECT = 0           ; Select frame – SETFRAME

                                    ; Select tool
$AC_MEAS_T_NUMBER = 1               ;
$AC_MEAS_D_NUMBER = 1               ;

$AC_MEAS_TYPE = 12                  ; Set measurement type to groove

                                    ; Perform calculation
RETVAL = MEASURE()                  ;

if RETVAL <> 0                      ;
setal(61000 + RETVAL)               ;
endif

$P_SETFRAME = $AC_MEAS_FRAME ;

$P_SETFR = $P_SETFRAME              ; Write system frame in data management

g1 x0 y0                            ; Approach the groove center

m30
```

**Measurement of an 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.

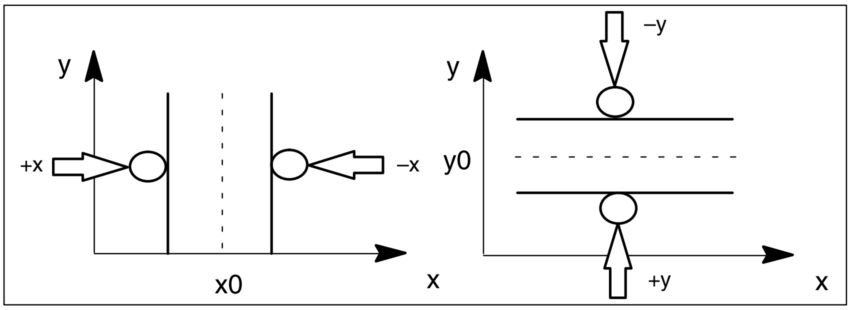



Fig. 2-13 Web

Table 2-25 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 1st point. *<br>0: +x, 1: –x, 2: +y, 3: –y, 4: +z, 5: –z |
| $AC_MEAS_TYPE | 13 |

* optional

Table 2-26 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 |

**Preset actual value memory for**

## Geo and special axes ($AC MEAS TYPE = 14)

This measurement type is used on the HMI operator interface

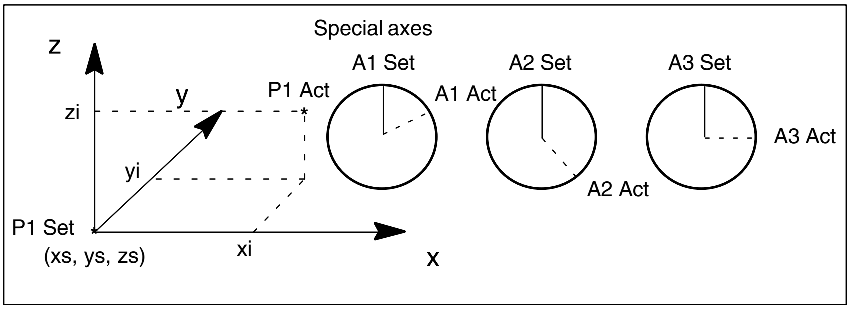



Fig. 2-14 Preset actual value memory for special axes only

Table 2-27 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 axes |
| $AA_MEAS_SETPOINT[axis] | Setpoint position of 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

Table 2-28 The following output variables are written for measurement type 14:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_FRAME | Result frame with translation |

**Preset actual value**

**For special axes only ($AC MEAS TYPE = 15)**

This measurement type is used on the HMI operator interface

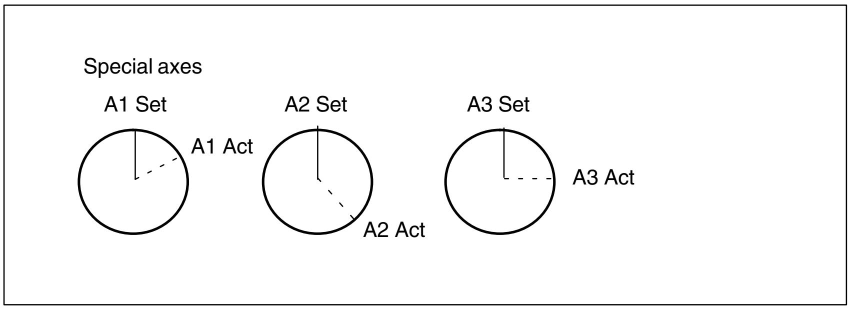



Fig. 2-15 Preset actual value memory

Table 2-29 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 axes |
| $AA_MEAS_SETPOINT[axis] | Setpoint position of 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

Table 2-30 The following output variables are written for measurement type 15:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_FRAME | Result frame with translation |

**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.

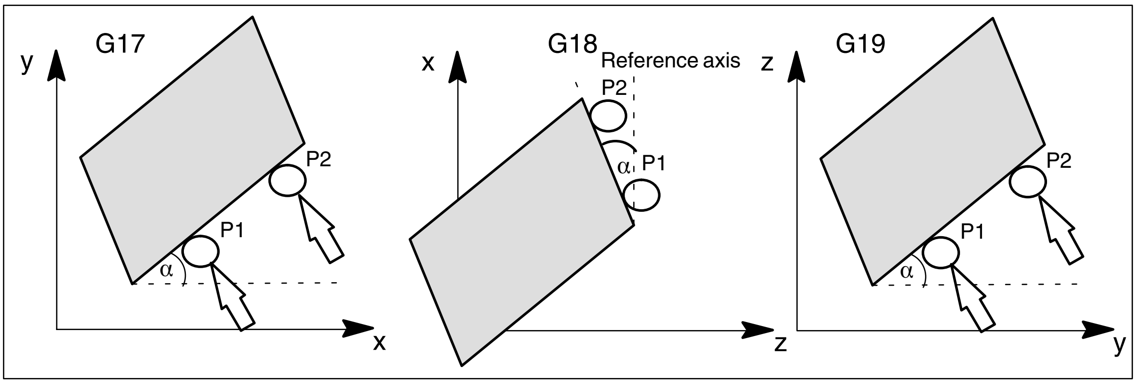



Fig. 2-16 Oblique edge

Table 2-31 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. *<br>=0: Reference coordinate is the abscissa<br>=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

Table 2-32 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 |

**Measurement of an**

**Angle in a plane ($AC MEAS TYPE = 17)**

The oblique plane is determined via three measuring points, P1, P2 and P3.

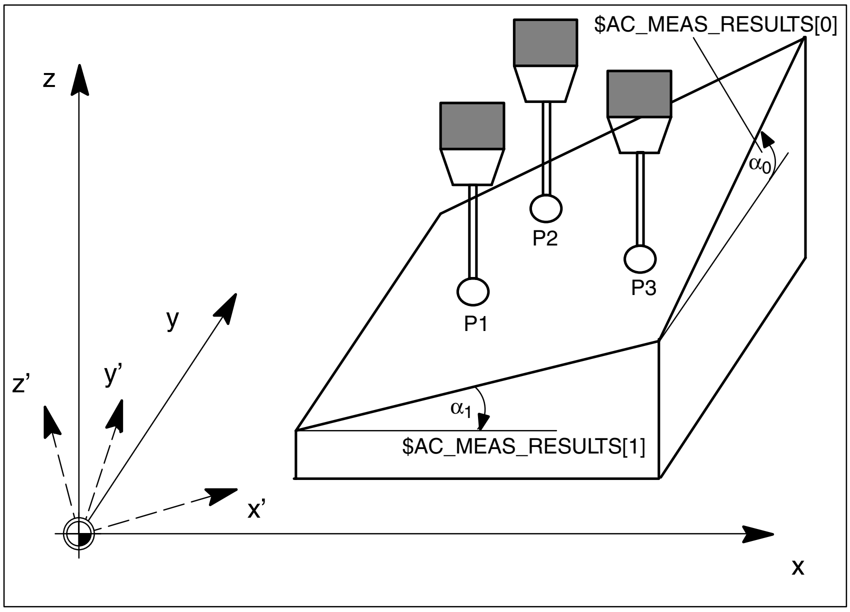



Fig. 2-17 Oblique plane in G17

$AC_MEAS_TYPE = 17 defines two resulting angles $\alpha_0$ and $\alpha_1$ for the skew of the plane; these are entered in $AC_MEAS_RESULTS[0..1]. In

- $AC_MEAS_RESULTS[0] contains the rotation about the abscissa and
- $AC_MEAS_RESULTS[1] the rotation about the ordinate.

These angles are calculated by means of the three measuring points P1, P2 and P3. With this measurement type, the angle in

- $AC_MEAS_RESULTS[2] for the applicate is always set to a default of 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.

Table 2-33 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 |

Table 2-34 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_SETPOINT[axis] | Setpoint rotations about abscissa and ordinate optional |
| $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 | 17 |

* optional

Table 2-35 The following output variables are written for measurement type 17:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_FRAME | Result frame |
| $AC_MEAS_RESULTS[0] | Angles about abscissa from which three measuring points are calculated |
| $AC_MEAS_RESULTS[1] | Angles about ordinate from which three measuring points are calculated |
| $AC_MEAS_RESULTS[2] | Angles about applicate from which three measuring points are calculated |
| $AC_MEAS_RESULTS[3] | Angle about abscissa which is entered in the result frame |
| $AC_MEAS_RESULTS[4] | Angle about ordinate which is entered in the result frame |
| $AC_MEAS_RESULTS[5] | Angle about applicate which is entered in the result frame |

**Angle measurement**

**Programming example**

```
DEF INT RETVAL                          ;
DEF AXIS _XX, _YY, _ZZ                  ;

T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54

$AC_MEAS_VALID = 0                      ; Invalidate all input values

$AC_MEAS_TYPE = 17                      ; Set measurement type for oblique plane
$AC_MEAS_ACT_PLANE = 0                  ; Plane for measurement is G17

_XX = $P_AXN1                           ; Define axes according to plane
_YY = $P_AXN2                           ;
_ZZ = $P_AXN3                           ;

G17 G1 _XX = 10 _YY = 10 F1000          ; Approach 1st measuring point
MEAS = 1 _ZZ = ...                      ;
                                        ;
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]      ; Assign measured value abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]      ; Assign measured value ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]      ; Assign measured value applicate

G1 _XX = 20 _YY = 10 F1000              ; Approach 2nd measuring point
MEAS = 1 _ZZ = ...                      ;
                                        ;
$AA_MEAS_POINT2[_xx] = $AA_MW[_xx]      ; Assign measured value abscissa
$AA_MEAS_POINT2[_yy] = $AA_MW[_yy]      ; Assign measured value ordinate
$AA_MEAS_POINT2[_zz] = $AA_MW[_zz]      ; Assign measured value applicate

G1 _XX = 20 _YY = 20 F1000              ; Approach 3rd measuring point
MEAS = 1 _ZZ = ...                      ;
                                        ;
$AA_MEAS_POINT3[_xx] = $AA_MW[_xx]      ; Assign measured value abscissa
$AA_MEAS_POINT3[_yy] = $AA_MW[_yy]      ; Assign measured value ordinate
$AA_MEAS_POINT3[_zz] = $AA_MW[_zz]      ; Assign measured value applicate

                                        ; Input setpoints for angles
$AA_MEAS_SETPOINT[_xx] = 12             ; Rotation about abscissa
$AA_MEAS_SETPOINT[_yy] = 4              ; Rotation about ordinate

$AC_MEAS_FRAME_SELECT = 102             ; Select target frame (G55)

                                        ; Select tool
$AC_MEAS_T_NUMBER = 1                   ;
$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

if $AC_MEAS_RESULTS[1] <> 4
setal(61000 + $AC_MEAS_RESULTS[1])
endif

$P_UIFR[2] = $AC_MEAS_FRAME             ; Write measurement frame to data management
(G55)

G55 G0 AX[_xx] = 10 AX[_yy] = 10        ; Activate frame and traverse
m30
```

**WCS reference on oblique plane**

**Redefine WCS ($AC MEAS TYPE = 18)**

The zero of the new WCS is defined by measuring point P1 at surface normal on the oblique plane.

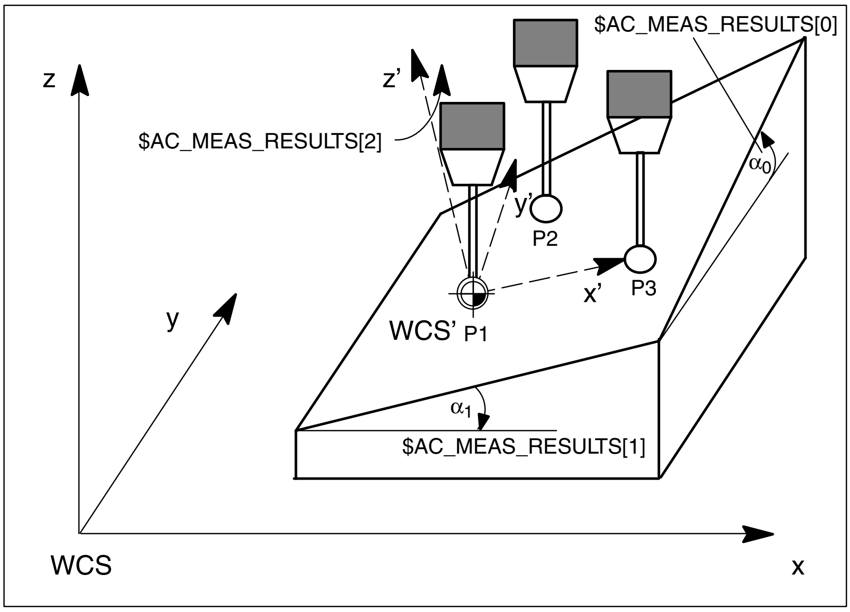



Fig. 2-18 Oblique plane in G17

**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 about the abscissa of the old WCS is stored in $AC_MEAS_RESULTS[0],
- The angle about the ordinate is stored in $AC_MEAS_RESULTS[1] and
- The angle about 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 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 in this plane. In this case,

- 1. measuring point P1 is applied as the new WCS' reference point,
- 2. measuring point P2 specifies the direction of abscissa x' of the new rotated WCS' coordinate system while
- 3. measuring point P3 is used to determine the solid angles.

Table 2-36 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) * |
| $AC_MEAS_D_NUMBER | Calculated as active D unless otherwise specified (D0) * |
| $AC_MEAS_TYPE | 18 |

* optional

Table 2-37 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 about the abscissa |
| $AC_MEAS_RESULTS[1] | Calculated angle about the ordinate |
| $AC_MEAS_RESULTS[2] | Calculated angle about the applicate |

**WCS reference on oblique plane**

**Programming example**

```
DEF INT RETVAL                          ;
DEF AXIS _XX, _YY, _ZZ                  ;

T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54

$AC_MEAS_VALID = 0                      ; Invalidate all input values

$AC_MEAS_TYPE = 18                      ; Set measurement type for oblique plane
$AC_MEAS_ACT_PLANE = 0                  ; Plane for measurement is G17

_XX=$P_AXN1                             ; Define axes according to plane
_YY=$P_AXN2                             ;
_ZZ=$P_AXN3                             ;

G17 G1 _XX=10 _YY=10 F1000              ; Approach 1st measuring point
MEAS = 1 _ZZ=...                        ;
                                        ;
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]      ; Assign measured value abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]      ; Assign measured value ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]      ; Assign measured value applicate

G1 _XX=20 _YY=10 F1000                  ; Approach 2nd measuring point
MEAS = 1 _ZZ=...                        ;
                                        ;
$AA_MEAS_POINT2[_xx] = $AA_MW[_xx]      ; Assign measured value abscissa
$AA_MEAS_POINT2[_yy] = $AA_MW[_yy]      ; Assign measured value ordinate
$AA_MEAS_POINT2[_zz] = $AA_MW[_zz]      ; Assign measured value applicate

G1 _XX=20 _YY=20 F1000                  ; Approach 3rd measuring point
MEAS = 1 _ZZ=...                        ;
                                        ;
$AA_MEAS_POINT3[_xx] = $AA_MW[_xx]      ; Assign measured value abscissa
$AA_MEAS_POINT3[_yy] = $AA_MW[_yy]      ; Assign measured value ordinate
$AA_MEAS_POINT3[_zz] = $AA_MW[_zz]      ; Assign measured value applicate

                                        ; Input setpoints for P1
$AA_MEAS_SETPOINT[_xx] = 10             ;
$AA_MEAS_SETPOINT[_yy] = 10             ;
$AA_MEAS_SETPOINT[_zz] = 10             ;

$AC_MEAS_FRAME_SELECT = 102             ; Select target frame (G55)

                                        ; Select tool
$AC_MEAS_T_NUMBER = 1                   ;
$AC_MEAS_D_NUMBER = 1                   ;

RETVAL = MEASURE()                      ; Start measurement calculation

if RETVAL <> 0
setal(61000 + RETVAL)
endif

                                        ; Calculation results of solid angles
                                                      ; Angle about the
R0 = $AC_MEAS_RESULTS[0]                              ; abscissa of old WCS
R1 = $AC_MEAS_RESULTS[1]                              ; ordinate
R2 = $AC_MEAS_RESULTS[2]                              ; applicate

$P_UIFR[2] = $AC_MEAS_FRAME             ; Write measurement frame to data management
(G55)

G55 G0 AX[_xx]=10 AX[_yy]=10            ; Activate frame and traverse
m30
```

**1-dimensional**

### Setpoint input ($AC MEAS TYPE = 19)

With this measurement method it is possible to input exactly one setpoint for the abscissa, the ordinate and the applicate. If two or three setpoints are input, 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.

Table 2-38 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 | Frame is written to coarse translation unless otherwise specified * |
| $AC_MEAS_TYPE | 19 |

* optional

Table 2-39 The following output variables are written for measurement type 19:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_FRAME | Result frame with rotations and translation |

**Setpoint input**

### Programming example

1-dimensional setpoint input

```
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              ; Invalidate all input values

$AC_MEAS_TYPE = 19              ; Set measurement type to 1-dimensional
                                ; setpoint input

$AC_MEAS_ACT_PLANE = 0          ; G17 is the measurement plane

_XX=$P_AXN1                     ; Define axes according to plane
_YY=$P_AXN2                     ;
_ZZ=$P_AXN3                     ;
```

```
                                        ; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]      ; Assign measured value abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]      ; Assign measured value ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]      ; Assign measured value applicate

$AA_MEAS_SETPOINT[_xx] = 10             ; Input 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 to data management
(G55)

G55 G0 AX[_xx]=10 AX[_yy]=10            ; Activate frame and traverse
m30
```

**2-dimensional**

### Setpoint input ($AC MEAS TYPE = 20)

Setpoints for two dimensions can be input with 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 pure actual value memory preset.

Table 2-40 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 | Frame is written to coarse translation unless otherwise specified * |
| $AC_MEAS_TYPE | 20 |

* optional

Table 2-41 The following output variables are written for measurement type 20:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_FRAME | Result frame with rotations and translation |

**Setpoint input**

### Programming example

2-dimensional setpoint input

```
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                          ; Invalidate all input values

$AC_MEAS_TYPE = 20                          ; Set measurement type to 2-dimensional
                                            ; setpoint input

$AC_MEAS_ACT_PLANE = 0                      ; G17 is the measurement plane

_XX=$P_AXN1                                 ; Define axes according to plane
_YY=$P_AXN2                                 ;
_ZZ=$P_AXN3                                 ;

                                            ; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]          ; Assign measured value abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]          ; Assign measured value ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]          ; Assign measured value applicate

$AA_MEAS_SETPOINT[_xx] = 10                 ; Input 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 to data management
(G55)

G55 G0 AX[_xx]=10 AX[_yy]=10                ; Activate frame and traverse
m30
```

**3-dimensional**

### Setpoint input ($AC MEAS TYPE = 21)

With this measurement method it is possible to input 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.

Table 2-42 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 | Frame is written to coarse translation unless otherwise specified * |
| $AC_MEAS_TYPE | 21 |

* optional

Table 2-43 The following output variables are written for measurement type 21:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_FRAME | Result frame with rotations and translation |

**Setpoint input**

### Programming example

3-dimensional setpoint input

```
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              ; Invalidate all input values

$AC_MEAS_TYPE = 21              ; Set measurement type to 3-dimensional
                                ; setpoint input

$AC_MEAS_ACT_PLANE = 0          ; G17 is the measurement plane

_XX=$P_AXN1                     ; Define axes according to plane
_YY=$P_AXN2                     ;
_ZZ=$P_AXN3                     ;
```

```
                                       ; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]              ; Assign measured value abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]              ; Assign measured value ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]              ; Assign measured value applicate

                                       ; Input setpoints for abscissa, ordinate and
$AA_MEAS_SETPOINT[_xx] = 10            ; applicate
$AA_MEAS_SETPOINT[_yy] = 10            ;
$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 to data management
                                       ; (G55)

G55 G0 AX[_xx]=10 AX[_yy]=10           ; Activate frame and traverse
m30
```

**TL with zoom-in function**

### Measurement of tool lengths 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.

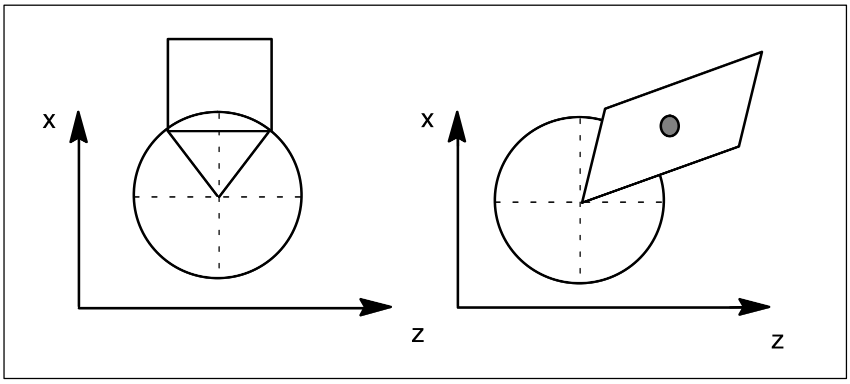



Fig. 2-19 Measurement of tool lengths with zoom-in function

Table 2-44 The values of the following variables are evaluated for measurement type 22:

| Input variable | Meaning |
|---|---|
| $AC_MEAS_VALID | Validity bits for input variables |
| $AA_MEAS_POINT1[axis] | Measuring point 1 for all channel axes |
| $AA_MEAS_P1_COORD | Coordinate system of measuring point * |
| $AA_MEAS_SETPOINT[axis] | Zoom positions x and z must be specified |
| $AA_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

Table 2-45 The following output variables are written for measurement type 22:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_RESULT[0] | Tool length in x, only if x0 is input |
| $AC_MEAS_RESULT[1] | Tool length in y, only if y0 is input |
| $AC_MEAS_RESULT[2] | Tool length in z, only if z0 is input |

**TL with stored / current position**

### Measuring a tool length 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.

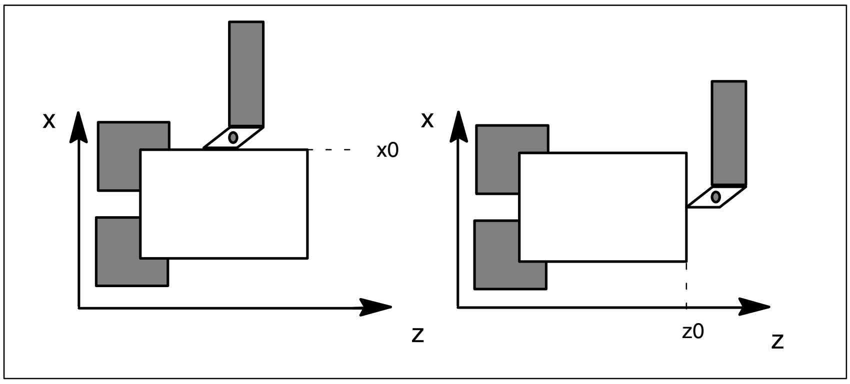



Fig. 2-20 Measurement of a tool length with a stored or actual position

Table 2-46 The values of the following variables are evaluated for measurement type 23:

| Input variable | Meaning |
|---|---|
| $AC_MEAS_VALID | Validity bits for input variables |
| $AA_MEAS_POINT1[axis] | Measuring point 1 or marked position |
| $AA_MEAS_P1_COORD | Coordinate system of measuring point * |
| $AA_MEAS_SETPOINT[axis] | Setpoint positions x0 and z0 must be specified |
| $AA_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 | 23 |

* optional

Table 2-47 The following output variables are written for measurement type 23:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_RESULT[0] | Tool length in x, only if x0 is input |
| $AC_MEAS_RESULT[1] | Tool length in y, only if y0 is input |
| $AC_MEAS_RESULT[2] | Tool length in z, only if z0 is input |

### Coordinate transformation of a position ($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.

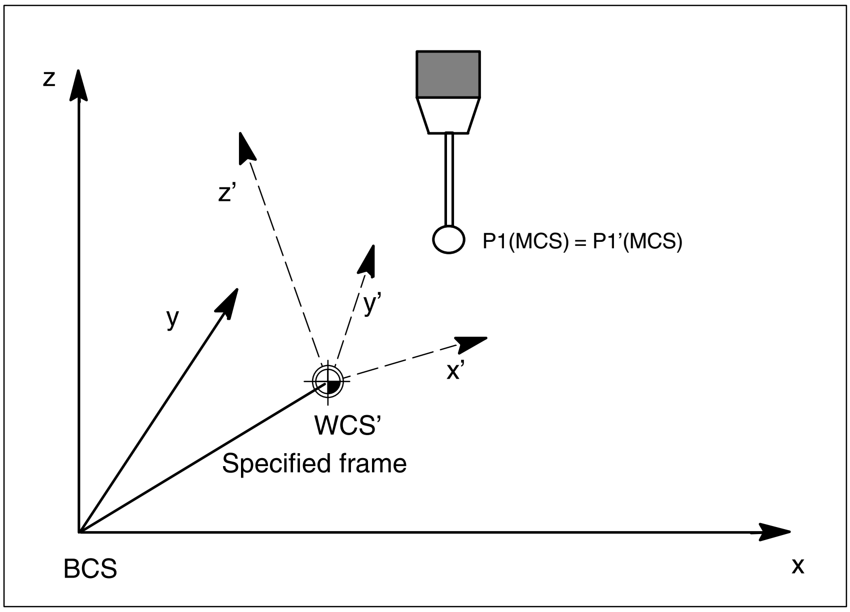



Fig. 2-21 Coordinate transformation of a position

Table 2-48 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 |
| $AA_MEAS_P1:COORD | Default is 0: WCS, 1: BCS, 2: MCS * |
| $AA_MEAS_P2_COORD | Target coordinate system * |
| $AA_MEAS_CHSFR | System frames from data management * |
| $AC_MEAS_NCBFR | Global basic frames from the data management * |
| $AC_MEAS_CHBF | Channel basic frames from the data management * |
| $AC_MEAS_UIFR | Settable frame from data management * |
| $AC_MEAS_PFRAME | Programmable frame is not included in calculation * |
| $AC_MEAS_TYPE | 24 |

* optional

Table 2-49 The following output variables are written for measurement type 24:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_POINT2[axis] | Converted axis positions |

### Rectangle ($AC MEAS TYPE = 25)

To determine a rectangle, tool dimensions are required in 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

with four measuring points 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.

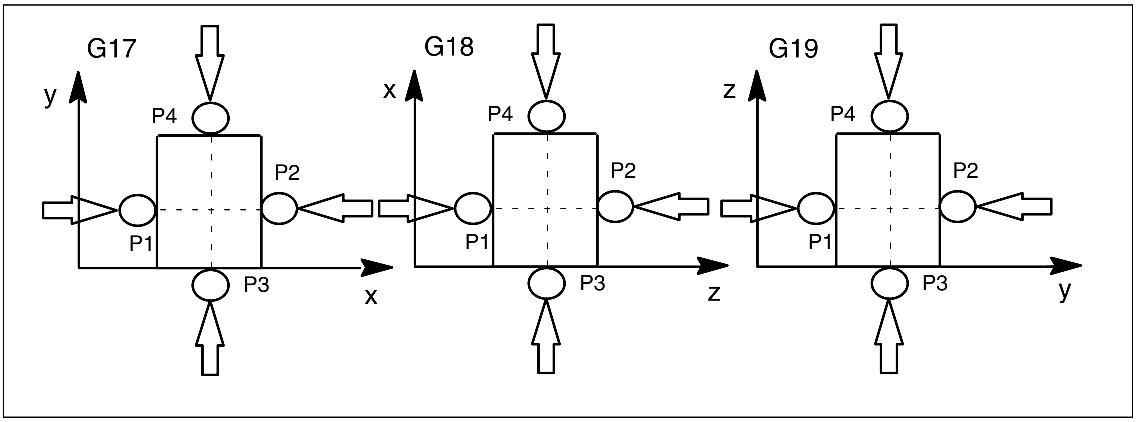



Fig. 2-22 Rectangle

| 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) * |
| $AC_MEAS_INPUT[0] | without specification of outer corner *<br>=0: Measurement for outer corner<br>=1: Measurement for inner corner |
| $AC_MEAS_TYPE | 25 |

* optional

Table 2-50 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 calculated center point |
| $AC_MEAS_RESULTS[1] | Ordinate of calculated center point |
| $AC_MEAS_RESULTS[2] | Applicate of calculated center point |
| $AC_MEAS_RESULTS[3] | Width of rectangle P1/P2 |
| $AC_MEAS_RESULTS[4] | Length of rectangle P3/P4 |

### Save 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 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_NCBFR = 0 channel basic frames;
- $AC_MEAS_UIFR = 0 number of settable frames

from the data management system by 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. Via the MD 18351: MM_DRAM_FILE_MEM_SIZE.

---

| Input variable | Meaning |
|---|---|
| $AC_MEAS_VALID | Validity bits for input variables |
| $AA_MEAS_CHSFR | Bit mask system frames from data management. *<br>If this variable is not written, all system frames are backed up. |
| $AA_MEAS_NCBFR | Bit mask of global basic frames from the data management. *<br>If this variable is not written, all global basic frames are backed up. |
| $AA_MEAS_CHBFR | Bit mask of channel basic frames from the data management. *<br>If this variable is not written, all channel basic frames are backed up. |
| $AA_MEAS_UIFR | Number of settable frames from data management 0..100: 1: G500 2: G500, G54. *<br>If this variable is not written, all settable frames are backed up. |
| $AC_MEAS_TYPE | 26 |

* optional

### Restore data management frames ($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 which has not been backed up is selected, the selection is ignored. The process is not aborted.

| Input variable | Meaning |
|---|---|
| $AC_MEAS_VALID | Validity bits for input variables |
| $AA_MEAS_CHSFR | Bit mask system frames from data management. *<br>If this variable is not written, all system frames are restored. |
| $AA_MEAS_NCBFR | Bit mask of global basic frames from the data management. *<br>If this variable is not written, all global basic frames are restored. |
| $AA_MEAS_CHBFR | Bit mask of channel basic frames from the data management. *<br>If this variable is not written, all channel basic frames are restored. |
| $AA_MEAS_UIFR | Number of settable frames from data management range from 1: G54 to G99: G599. * If this variable is not written, all settable frames are restored. |
| $AC_MEAS_TYPE | 27 |

* optional

## 2.3.3 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.

**Measurement of**

**Tool length ($AC MEAS TYPE = 10)**

The tool length 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.

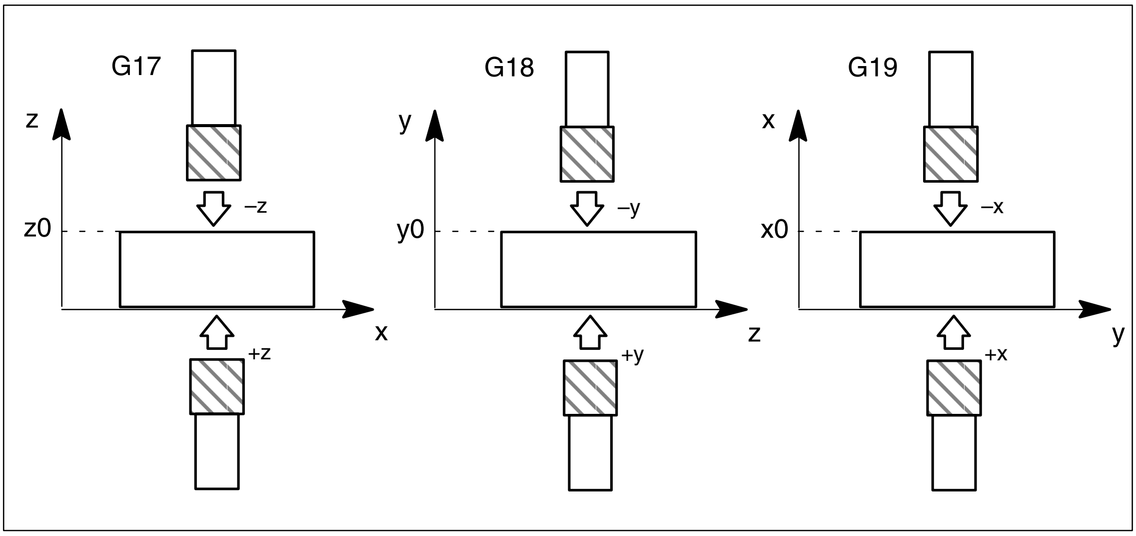



Fig. 2-23 Tool length

Table 2-51 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 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 |
| $AC_MEAS_ACT_PLANE | Calculated as active plane unless otherwise specified * |
| $AC_MEAS_TYPE | 10 |

* optional

Table 2-52 The following output variables are written for measurement type 10:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_TOOL_LENGTH | Tool length |

**Measure tool length**

### Programming example

```
DEF INT RETVAL                     ;

T0 D0                              ;
g0 x0 y0 z0 f10000                 ;

$AC_MEAS_VALID = 0                 ; Invalidate all input values

g1 z10                             ; Move tool to reference part
$AC_MEAS_LATCH[0] = 1              ; Pick up measuring point 1

$AC_MEAS_DIR_APPROACH = 5          ; Set approach direction –z

                                   ; Set reference position
$AA_MEAS_SETPOINT[x] = 0           ;
$AA_MEAS_SETPOINT[y] = 0           ;
$AA_MEAS_SETPOINT[z] = 0           ;

$AC_MEAS_ACT_PLANE = 0             ; G17 is the measurement plane

                                   ; No tool is selected
$AC_MEAS_T_NUMBER = 1              ;
$AC_MEAS_D_NUMBER = 1              ;

$AC_MEAS_TYPE = 10                 ; Set measurement type to tool length

                                   ; Perform calculation
RETVAL = MEASURE()                 ;

if RETVAL <> 0
setal(61000 + RETVAL)
endif

if $AC_MEAS_TOOL_LENGTH <> 10      ; Scan known tool length
setal(61000 + $AC_MEAS_TOOL_LENGTH)
endif

m30
```

**Measurement of**

**Tool diameter ($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.

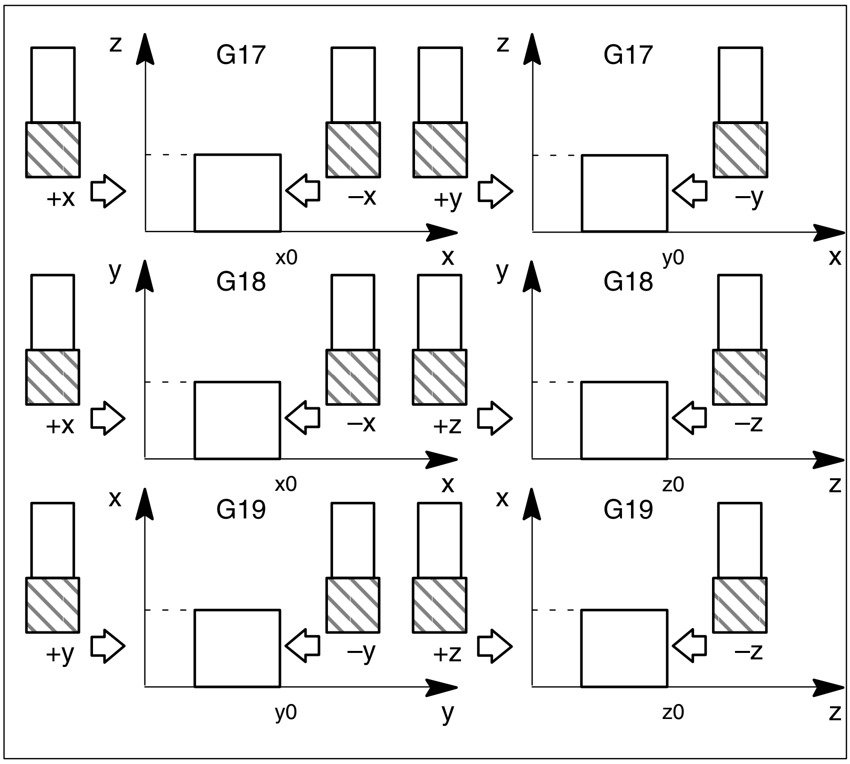



Fig. 2-24 Tool diameter for selected planes G17, G18 and G19

Table 2-53 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 |
| $AA_MEAS_SETPOINT[axis] | Set position x0 |
| $AC_MEAS_ACT_PLANE | Calculated as active plane unless otherwise specified * |
| $AC_MEAS_TYPE | 11 |

* optional

Table 2-54 The following output variables are written for measurement type 11:

| Output variable | Meaning |
|---|---|
| $AC_MEAS_TOOL_DIAMETER | Tool diameter |

# 2.4 Axial measurement (option)

A measuring operation can be initiated from both the part program and synchronized actions. A measuring mode, the encoder and up to four trigger events are programmed, the trigger events comprising the probe number (1 or 2) and the activation criterion (rising/falling signal edge).

If the measured values are to be stored from encoder 1 and 2 for each trigger event, then only two trigger events can be programmed.

## 2.4.1 Software requirements

**NC software version**

Axial measuring functionality is available as an option.

**Operator panel front software versions**

The "Measurement result display" and "Parameter assignment" functions are available via input dialog with

- MMC software version
- HMI Advanced and HMI Embedded

## 2.4.2 Supplementary conditions

**Mode change**

**Measuring job from part program**
A measuring job activated by a part program is not affected by a change in operating mode. However, it is deleted immediately the program advances to a new block.
RESET aborts the measuring jobs.

**Measuring job from synchronized actions**
A measuring job activated by a modal synchronized action is not affected by a changeover in operating mode. The job is modally active beyond block limits.

**Block search**

**Measuring job from part program**
The job is not started. No measurement check-back signals are supplied.

**Measuring job from synchronized actions**
Modal measuring jobs are not activated until the programmed conditions are fulfilled.

**Repos**

**Measuring job from part program**
If a measuring job is currently in progress, it is aborted and restarted again after the REPOS block. If the job had already been completed, it is not started again.

**Measuring job from synchronized actions**
Activated jobs remain unaffected.

## 2.4.3 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.

**Measuring mode 1**

The user can program up to four different trigger events in the same position control cycle.
The measuring signal edges are evaluated in chronological sequence.

- Up to two probes with two measuring signal edges each can be programmed for each measurement job. If 2 encoders are used, the number of programmed trigger events is halved.
- Where six-axis modules are installed, measuring mode 1 is imaged on measuring mode two internally in the control.
- The traversing velocity must be lower or equal to the shortest distance between two identical trigger events in each position control clock cycle.

---

**Note**

With this mode, the compensation value which was present when the last measuring signal edge was received is calculated for all measured values.

---

**Measuring mode 2**

The user can program up to four different trigger events one after the other in the configured sequence.
Evaluation of measuring signal edges is activated for one trigger event at a time and takes place in the programmed sequence.

- Trigger events are detected only in the programmed sequence.
- The traversing velocity must be lower or equal to the shortest distance between two trigger events in each position control clock cycle.

---

**Note**

**Measurement does not work on simulated axes!**

---

**Measuring probe status**

It is possible to scan the probe status directly in the parts program and in synchronized actions.

$A_PROBE[n] with n=probe
$A_PROBE[n]==1: probe deflected
$A_PROBE[n]==0: probe not deflected

## 2.4.4 Programming

**Programming**

Axial measurements can be programmed with and without deletion of distance-to-go.

MEASA with deletion of distance-to-go
MEAWA without deletion of distance-to-go

MEASA[Achse] = (mode, trigger event1, trigger event2, trigger event3, trigger event4)

**Parameter description:**

- Axis : Channel axis name (X, Y, ...)
- Mode : Ones decade
  0 = Abort measurement job (e.g. for synchronized actions)
  1 = Up to four trigger events that can be activated simultaneously
  2 = Up to four trigger events that can be activated sequentially
  Error output if the first trigger event is already active
  3 = Up to four trigger events that can be activated sequentially
  **NONE** Error output if the first trigger event is already active, alarms 21700/21703 are suppressed

  Tens decade (= encoder selection)
  0/not set = Use active measuring system
  1 = 1st measuring system
  2 = 2nd measuring system (If installed. Otherwise the first measuring system is used, no alarm is output)
  3 = 1st and 2nd measuring system

  If the measurement is taken using two measuring systems, a maximum of two trigger events may be programmed. The measured values of both encoders are recorded for each of the two trigger events.
- Trigger event 1 = rising edge of probe 1
  –1 = falling edge of probe 1
  2 = rising edge of probe 2
  –2 = falling edge of probe 2

---

**Note**

MEASA and MEAWA can be programmed in the same block.

MEASA cannot be programmed in synchronized actions.

The axes for which MEASA has been programmed are not decelerated until all programmed trigger events have arrived.

Measurement jobs started from a parts program are aborted by RESET or when the program advances to a new block.

If MEASA/MEAWA are programmed in the same block as MEAS/MEAW, the configuration is rejected with alarm 21701.

If a geometry axis is used in a measurement, the measured values are only made available in the workpiece coordinate system if all geometry axes are programmed with the same measurement task. If a geometry axis is missing from the measurement task, the measured values are stored only in the machine coordinate system and alarm 21702 is output. The same applies to axes that are involved in a transformation.
If the measurement must start on the probe signal edge (with the position of the probe unknown at the instant measurement commences), the customer must evaluate the probe in the parts program. By scanning the probe status, it is generally possible to ensure that the next probe signal edge (positive or negative) detected in the hardware will initiate the measurement job.

```
if $A_PROBE [1] = 1         ; Probe deflected?
MEAC [X] = (1,1,–1,1)       ; Starts on the first detected negative edge.
else
MEAC [X] =(1,1,1,–1)        ; Starts on the first detected positive edge.
endif.
```

The alarms are described in the online help or in
**References** /DA/ Diagnostic Guide

---

## 2.4.5 Measurement results

**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 parts program.

- **System variable $AC_MEA[No]**
  Scan status signal of measurement job.
  <[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 parts program.

- **System variable $AA_MM1[axis]** to **$AA_MM4[axis]**
  Access to measurement result of trigger signal in machine coordinate system. Read in part program and in synchronized actions.
  <Axis> stands for the name of the measurement axis (X, Y, ...).

- **System variable $AA_MM1[axis]** to **$AA_MM4[axis]**
  Access to measurement result of trigger signal in machine coordinate system. Read in part program and in synchronized actions.
  <Axis> stands for the name of the measurement axis (X, Y, ...).

Programming

If two measuring systems are used to take the measurement, a maximum of two trigger events may be programmed. The measured values of both encoders are recorded for each of the two trigger events.

**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**

The functional test for the probe is conducted via an NC program.

The measuring signal can be checked at the end of the program in the diagnostic menu “PLC status”.

Table 2-55 Status display for measuring signal

| | Status display |
|---|---|
| Probe 1 deflected | DB10 DB B107.0 |
| Probe 2 deflected | DB10 DB B107.1 |

**References:** /PAZ/, Programming Guide Cycles
/BNM/, User’s Guide Measuring Cycles

### 2.4.6 Continuous measurement (cyclic measurement)

All measurements are written to a previously defined FIFO variable. The number of measured values is defined in machine data.

- Correct operation of the function can only be relied upon with an IPO/position control ratio of $\leq$ 8 : 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.

**MEAC**

Continuous, axial measurements without deletion of distance-to-go

MEAC[axis] = (mode, measurement memory, trigger event 1,
trigger event 2, trigger event 3,
trigger event 4)

**Parameter description:**

- Axis : Channel axis name (X, Y, ...)
- Mode : Ones decade
  0 = Abort measurement job (for synchronized actions)
  1 = Up to four trigger events that can be simultaneously activated (a maximum of four signals can be triggered simultaneously in one position controller cycle, but the correct order must be observed)
  2 = Up to four trigger events that can be activated sequentially (only **one** signal can be triggered per position controller cycle)

  Tens decade (= encoder selection)
  0/not enabled = active measuring system
  1 = 1st measuring system
  2 = 2nd measuring system (if installed, the first measuring system is otherwise used, no alarm is generated)
  3 = 1st and 2nd measuring system

  If the measurement is taken using two measuring systems, a maximum of two trigger events may be programmed.
- Measurement memory : Number of FIFO
- Trigger event 1 = rising edge of probe 1
  –1 = falling edge of probe 1
  2 = rising edge of probe 2
  –2 = falling edge of probe 2

The axial measurement values are available in the machine coordinate system (MCS). They are written to a FIFO variable defined by the user, e.g. $AC_FIFO1. When two probes are configured to take the measurement, the measured values from the second probe are stored separately in the following FIFO.
The number of measured values is limited by MD 28264: LEN_AC_FIFO. Variables $AC_MEA and $AA_MM are therefore irrelevant.

The values can be read from the FIFO both in the parts program and from synchronized actions.

The measurement is active until

- MEAC[“axis”]=(0) is programmed
- a FIFO is full
- RESET is pressed or end of program M02/M30 is detected

**Endless measuring**

In order to implement endless measuring, FIFO values must be read cyclically from the part program. The frequency at which measured values are read from the FIFO memory and processed must correspond to the write rate of the NC. The number of valid entries can be read in a FIFO variable.
In order to achieve a defined number of measured values, the measuring function must be explicitly deselected by the program.

**FIFO variables**

For definition of FIFO variables, see
**References** /FB2/, S5, Synchronized Actions

## 2.5 Measurement accuracy and functional testing

### 2.5.1 Measuring accuracy

**Accuracy**

The propagation time of the measuring signal is determined by the hardware used. The delay times when using SIMODRIVE 611D are in the 3.625 μ ... 9.625 μ range plus the reaction time of the probe.

The measurement uncertainty is calculated as follows:
Measurement uncertainty = measuring signal propagation time x traversing velocity

The permissible traversing velocities depend on the number of programmed measuring signal edges and the ratio between the IPO clock cycle and position control cycle.
Accurate measuring results are obtained only at traversing velocities at which no more than 1 identical and no more than 4 different trigger signals arrive in each position control cycle.

### 2.5.2 Probe functional test

**Example of functional test**

```
%_N_TEST_PROBE_MPF
;$PATH=/_N_MPF_DIR
;Testing program probe connection
N05   DEF INT MTSIGNAL                ;Flag for trigger status

N10   DEF INT ME_NR=1                 ;Measuring input number
N20   DEF REAL MEASVALUE_IN_X
N30   G17 T1 D1                       ;Preselect tool
                                      ;Preselect probe
N40   _ANF: G0 G90 X0 F150            ;Start position and
                                      ;measuring velocity
N50   MEAS=ME_NR G1 X100              ;Measurement at measuring
                                      ;input 1 in X axis
N60   STOPRE
N70   MTSIGNAL=$AC_MEA[1]             ;Read software switching signal
                                      ;at 1st measuring input
N80   IF MTSIGNAL == 0 GOTOF _FEHL1   ;Evaluate signal
N90   MEASVALUE_IN_X=$AA_MW[X]        ;Read measured value
                                      ;in workpiece coordinates
N95   M0
N100  M02
N110  _FEHL1: MSG (“Probe not switching!”)
N120  M0
N130  M02
```

■

# Notes

# Supplementary Conditions 3

Axial measurement functionality is available with SW package 4 and higher.

The function is not contained in the export version SINUMERIK 840DE/810DE.

■

# Notes

# 4 Data Descriptions (MD, SD)

| | | | | | |
|---|---|---|---|---|---|
| **13200**<br>MD number | **MEAS_PROBE_LOW_ACTIVE**<br>Switching characteristics of probe | | | | |
| Default setting: FALSE<br>**When using measurement cycles: FALSE** | | Minimum input limit: FALSE | | Maximum input limit: TRUE | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: – |
| Data type: BOOLEAN | | | | Applies from SW: SW 2.2 | |
| Meaning: | Value 0: (default setting)<br>non-deflected state 0 V (Low)<br>deflected state 24 V (High)<br>Value 1: non-deflected state 24 V (High)<br>deflected state 0 V (Low) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **13201**<br>MD number | **MEAS_PROBE_SOURCE**<br>Measurement pulse simulation via digital output | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 8 | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: SW 6.1 | |
| Meaning: | Function value on host only<br>Value > 0: Measurement pulse simulation starts with the digital output.<br>Value = 1: Measurement pulse simulation starts immediately after the measurement job. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **13210**<br>MD number | **MEAS_TYPE**<br>Type of measurement for PROFIBUS-DP drives | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: SW 6.1 | |
| Meaning: | This machine data is used to set the measurement function for distributed drives. This machine data is currently only functional for PROFIBUS-DP drives.<br><br>MD13210: MEAS_TYPE<br>Value = 0: A centralized probe connected to the NC is used. However, since only cyclic position values are supplied by the encoders, the actual measurement position is determined by interpolation.<br><br>MD13210: MEAS_TYPE<br>Value = 1: A distributed probe must be connected to **ALL** drives. The measurement functionality of the drives is then used. The actual encoder values in the hardware at the time of the measuring signal edge are stored. This method is more accurate than MD 13210: MEAS_TYPE = 0, but requires additional wiring and drives which support this measurement functionality, such as SIMODRIVE 611 universal. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **28264**<br>MD number | **LEN_AC_FIFO**<br>Length of $AC_FIFO ... FIFO variables | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 10000 | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: SW 4.1 | |
| Meaning: | Length of FIFO variables $AC_FIFO1 to $AC_FIFO10.<br>All FIFO variables are equal in length. | | | | |

■

# 5 Signal Descriptions

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX62.3**<br>Data block | **Measuring status**<br><br>Signal(s) from axis/spindle (NCK → PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 4 |
| Signal state 1 or signal transition 0 ——> 1 | The “Measuring” function is active.<br>This signal is used during measuring and displays the current measuring status of the axis. | | |
| Signal state 0 or signal transition 1 ——> 0 | The “Measuring” function is not active. | | |

| | | | |
|---|---|---|---|
| **DB10, ...**<br>**DBX107.0 and 107.1**<br>Data block | **Probe actuated**<br><br>Signal(s) from axis/spindle (drive → PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | Probe 1 or 2 is actuated. | | |
| Signal state 0 or signal transition 1 ——> 0 | Probe 1 or 2 is not actuated. | | |
| References | /PHD/, “NCU 571 – 573 Manual”<br>/PHF/, “NCU 570 Manual” | | |
| Note | With SW 3.2 and earlier, the signal is active only while the NC block containing the measuring operation is being processed. | | |

■

# Notes

# Example 6

## 6.1 Measuring mode 1

**Measurement with one encoder**

– Single measurement
– One probe
– Trigger signals are the rising and falling edges
– Actual value from current encoder

```
N2   MEASA[X] = (1, 1, –1) G01 X100 F100
N3   STOPRE
N4   IF $AC_MEA[1]==FALSE gotof END
N5   R10=$AA_MM1[X]
N6   R11=$AA_MM2[X]
N7   END:
```

**Measurement with two encoders**

– Single measurement
– One probe
– Trigger signals are the rising and falling edges
– Actual values from two encoders

```
N2   MEASA[X]=(31, 1, –1) G01 X100 F100
N3   STOPRE
N4   IF $AC_MEA[1]==FALSE gotof END
N5   R10=$AA_MM1[X]
N6   R11=$AA_MM2[X]
N7   R12=$AA_MM3[X]
N8   R13=$AA_MM4[X]
N9   END:
```

## 6.2 Measuring mode 2

– Two probes
– Trigger signals are the rising and falling edges
– Actual value from current encoder

```
N2    MEASA[X] = (2, 1, –1, 2, –2) G01 X100 F100
N3    STOPRE
N4    IF $AC_MEA[1]==FALSE gotof PROBE2
N5    R10=$AA_MM1[X]
N6    R11=$AA_MM2[X]
N7    PROBE2
N8    IF $AC_MEA[2]==FALSE gotof END
N9    R12=$AA_MM3[X]
N10   R13=$AA_MM4[X]
N11   END:
```

## 6.3 Continuous measurement

### 6.3.1 Cont. measurement on completion of progr. traversing motion

– The measurement is taken in measuring mode 1
– Measurement with 100 values
– One probe
– Trigger signal is the falling edge
– Actual value from current encoder

```
N1    DEF REAL MEASVALUE[100]
N2    DEF INT INDEX=0
N3    MEAC[x]=(1, 1, –1) G01 X1000 F100
N4    MEAC[X]=(0)                        ;Abort
N5    R1=$AC_FIFO1[4]                    ;No. of measured values
N6    FOR INDEX=0 TO R1
N7    MEASVALUE[INDEX]=$AC_FIFO1[0]      ;Read out measured values
N8    ENDFOR:
```

### 6.3.2 Continuous measurements with deletion of distance-to-go

– Delete distance-to-go after last measurement
– The measurement is taken in measuring mode 1
– Measurement with 100 values
– One probe
– Trigger signal is the falling edge
– Actual value from current encoder

```
N1   DEF INT NUMBER=100
N2   DEF REAL MEASVALUE[NUMBER]
N3   DEF INT INDEX=0
N4   WHEN $AC_FIFO1[4]==NUMBER DO DELDTG (X) MEAC[X] =(0)
N5   MEAC[X]=(1, 1, –1) G01 X1000 F100        ;Start measurement
N6   R1=$AC_FIFO1[4]                          ;No. of measured values
N7   FOR INDEX=0 TO R1
N8   MEASVALUE[INDEX]=$AC_FIFO1[0]            ;Read out measured values
N9   ENDFOR:
```

### 6.3.3 Continuous measurements modally over several blocks

– The measurement is taken in measuring mode 1
– Measurement with 100 values
– One probe
– Trigger signal is the falling edge
– Actual value from current encoder

```
N1   DEF INT NUMBER=100
N2   DEF REAL MEASVALUE[NUMBER]
N3   DEF INT INDEX=0
N4   ID=1 MEAC[X]=(1, 1, –1)                 ;Start measurement
N5   ID=2 WHEN $AC_FIFO1[4]==NUMBER DO MEAC[X]=(0) CANCEL(2)
N6   G01 X1000 Y100
N7   X100 Y100
N8   R1=$AC_FIFO1[4]                         ;Number of measured values
N9   FOR INDEX=0 TO R1
N10  MEASVALUE[INDEX]=$AC_FIFO1[0]           ;Read out measured values
N11  ENDFOR:
```

## 6.4 Functional test and repeat accuracy

**Function test**

```
%_N_TEST_PROBE_MPF
;$PATH=/_N_MPF_DIR
;Testing program probe connection
N05   DEF INT MTSIGNAL                ;Flag for trigger status

N10   DEF INT ME_NR=1                 ;Measuring input number
N20   DEF REAL MEASVALUE_IN_X
N30   G17 T1 D1                       ;Preselect tool
                                      ;Preselect probe
N40   _ANF: G0 G90 X0 F150            ;Start position and
                                      ;measuring velocity
N50   MEAS=ME_NR G1 X100              ;Measurement at measuring
                                      ;input 1 in X axis
N60   STOPRE
N70   MTSIGNAL=$AC_MEA[1]             ;Read software switching signal
                                      ;at 1st measuring input
N80   IF MTSIGNAL == 0 GOTOF _FEHL1   ;Evaluate signal
N90   MEASVALUE_IN_X=$AA_MW[X]        ;Read measured value
                                      ;in 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 so-called random dimensional deviations which are not subject to any trend.

```
%_N_TEST_ACCUR_MPF;
$PATH=/_N_MPF_DIR
N05   DEF INT SIGNAL, II                ;Variable definition
N10   DEF REAL MEASVALUE_IN_X[10]
N15   G17 T1 D1                         ;Initial conditions,
                                        ;Preselect tool
                                        ;offset for probe
N20   _ANF: G0 X0 F150   ←              ;Preposition in measurement axis
N25   MEAS=+1 G1 X100  ←                ;Measurement at 1st measuring
                                        ;input with switching signal not
                                        ;deflected, deflected in the X axis
N30   STOPRE             ←              ;Stop decoding for subsequent
                                        evaluation of result
N35   SIGNAL= $AC_MEA[1]                ;Read software switching signal at
                                        ;1st measuring input
N37   IF SIGNAL == 0 GOTOF_FEHL1        ;Check switching signal
N40   MEASVALUE_IN_X[II]=$AA_MW[X]      ;Read measured value into
                                        ;workpiece coordinates
N50   II=II+1
N60   IF II<10 GOTOB_ANF                :Repeat ten times
N65   M0
N70   M02
N80   _FEHL1: MSG ("Probe is 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.

■

## Notes

# 7

# Data Fields, Lists

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ...)** | | | |
| 13200 | MEAS_PROBE_LOW_ACTIVE | Switching characteristics of probe | |
| 13201 | MEAS_PROBE_SOURCE | Measurement pulse simulation via digital output | |
| 13210 | MEAS_TYPE | Type of measurement for PROFIBUS DP drives | |
| **Channel-specific ($MC_...)** | | | |
| 20360 | TOOL_PARAMETER_DEF_MASK | Definition of tool parameters | W1 |
| 28264 | MM_LEN_AC_FIFO | Length of $AC_FIFO ... FIFO variables | |

## 7.1 System variable

Table of all the input values:

| Type | System variable name | Values | Description | PGA 1 |
|---|---|---|---|---|
| INT | $AC_MEAS_SEMA | 0: Unassigned<br>1: assigned to the cycle | Interface assignment | Measurement |
| INT | $AC_MEAS_VALID | Bit mask | Validity bits for input values | Measurement |
| REAL | $AA_MEAS_POINT1[axis] | mm | 1. Measuring point for all channel axes | Repos |
| REAL | $AA_MEAS_POINT2[axis] | Inch | 2. Measuring point for all channel axes | Repos |
| REAL | $AA_MEAS_POINT3[axis] | | 3. Measuring point for all channel axes | Repos |
| REAL | $AA_MEAS_POINT4[axis] | | 4. Measuring point for all channel axes | Repos |
| REAL | $AA_MEAS_SETPOINT[axis] | | Setpoint position for all channel axes | Repos |
| REAL | $AA_MEAS_SETANGLE[axis] | Degrees | Setpoint angle for all channel axes | Repos |
| INT | $AC_MEAS_P1_COORD | 0: WCS | Coord. system for the 1st measuring point | Measurement |
| INT | $AC_MEAS_P2_COORD | 1: BCS | Coord. system for the 2nd measuring point | Measurement |
| INT | $AC_MEAS_P3_COORD | 2: MCS | Coord. system for the 3rd measuring point | Measurement |
| INT | $AC_MEAS_P4_COORD | | Coord. system for the 4th measuring point | Measurement |
| INT | $AC_MEAS_SET_COORD | | Coordinate system of setpoint | Measurement |
| INT | $AC_MEAS_LATCH[0..3] | 0: Invalid | Pick up measuring points in the WCS | Measurement |
| INT | $AA_MEAS_P1_VALID[axis] | 1: Pickup | 1. Pick up measuring point in the WCS | Repos |
| INT | $AA_MEAS_P2_VALID[axis] | | 2. Pick up measuring point in the WCS | Repos |
| INT | $AA_MEAS_P3_VALID[axis] | | 3. Pick up measuring point in the WCS | Repos |
| INT | $AA_MEAS_P4_VALID[axis] | | 4. Pick up measuring point in the WCS | Repos |

| Type | System variable name | Values | Description | PGA 1 |
|---|---|---|---|---|
| INT | $AA_MEAS_SP_VALID[axis] | 0: Invalid<br>1: Valid | Set setpoint position of axis as valid | Repos |
| REAL | $AC_MEAS_WP_SETANGLE | [ –90, 90 ] | Setpoint workpiece position angle | Measurement |
| REAL | $AC_MEAS_CORNER_SETANGLE | [ 0, 180 ] | Setpoint cutting angle of corner | Measurement |
| INT | $AC_MEAS_DIR_APPROACH | 0: +x<br>1: –x<br>2: +y<br>3: –y<br>4: +z<br>5: –z | Approach direction | Measurement |
| INT | $AC_MEAS_ACT_PLANE | 0: G17<br>1: G18<br>2: G19 | G17 working plane x/y infeed direction z<br>G18 working plane z/x infeed direction y<br>G19 working plane y/z infeed direction x | Measurement |
| INT | $AC_MEAS_SCALEUNIT | 0: configured<br>1: act. G code | Unit of measurement INCH / METRIC | Measurement |
| INT | $AC_MEAS_FINE_TRANS | 0: coarse<br>1: fine | Corrections in fine displacement | Measurement |
| INT | $AC_MEAS_FRAME_SELECT | 0<br>10 .. 25<br>50 .. 65<br>100 .. 199<br>1010 .. 1065<br>1050 .. 1065<br>2000<br>2010 .. 2025<br>2050 .. 2065<br>2100 .. 2199<br>3010 .. 3025<br>3050 .. 3065 | $P_SETFRAME<br>$P_CHBFRAME[0..15]<br>$P_NCBFRAME[0..15]<br>$P_IFRAME<br>$P_CHBFRAME[0..15] / G500<br>$P_NCBFRAME[0..15] / G500<br>$P_SETFR<br>$P_CHBFR[0..15]<br>$P_NCBFR[0..15]<br>$P_UIFR[0..99]<br>$P_CHBFR[0..15] / G500<br>$P_NCBFR[0..15] / G500 | Measurement |
| INT | $AC_MEAS_CHSFR | 0 .. 7F | Frame chain setting: System frames | Measurement |
| INT | $AC_MEAS_NCBFR | 0 .. FFFF | Frame chain setting:<br>Global basic frames | Measurement |
| INT | $AC_MEAS_CHBFR | 0 .. FFFF | Frame chain setting:<br>Channel basic frames | Measurement |
| INT | $AC_MEAS_UIFR | 0 .. 99 | Frame chain setting:<br>Settable frames | Measurement |
| INT | $AC_MEAS_PFRAME | 0: in<br>1: out | Frame chain setting:<br>Prog. frame | Measurement |
| INT | $AC_MEAS_T_NUMBER | | Tool selection | Measurement |
| INT | $AC_MEAS_D_NUMBER | | Cutting edge selection | Measurement |
| INT | $AC_MEAS_TOOL_MASK | Bit mask | Tool settings | Measurement |
| INT | $AC_MEAS_TYPE | | Measuring type | Measurement |

Table of all the output values:

| Type | System variable name | Values | Description | PGA 1 |
|---|---|---|---|---|
| FRAME | $AC_MEAS_FRAME | | Result frame | Measurement |
| REAL | $AC_MEAS_WP_ANGLE | | Calculated workpiece position angle | Measurement |
| REAL | $AC_MEAS_CORNER_ANGLE | | Calculated angle of intersection | Measurement |
| REAL | $AC_MEAS_DIAMETER | | Calculated diameter | Measurement |
| REAL | $AC_MEAS_TOOL_LENGTH | | Calculated tool length | Measurement |
| REAL | $AC_MEAS_RESULTS[10] | | Calculation results (type-dependent) | Measurement |

■

## Notes

# SINUMERIK 840D/840Di/810D Description of Functions Extended Functions (FB2)

# Software Cams, Position Switching Signals (N3)

■

# Brief Description 1

The “Software cams” function can be used to output position-dependent cam signals to the PLC and to the NCK I/O devices in the position control cycle or according to a timer.

The position values at which the signal outputs are set can be defined and altered via setting data.

Thirty-two software cam pairs are available. These can be used, for example:

- As reversing signals for hydraulically controlled oscillation axes.
- As limit switches

As of SW 6.3:

- Provision of a pulse of a defined duration when a cam position is crossed (position-time cam) for optional evaluation
- Settable signal inversion for modulo rotary axes in cases where: Plus cam – minus cam > 180 degrees
- Gated output to the NCK I/O devices
- Timer-controlled cam signal output independent of interpolation cycle, without output priority for the onboard outputs (more precise resolution of cam edges within interpolation cycle).

---

**Note**

Software cams can be applied for linear axes and modulo rotary axes.

---

■

## Notes

# Detailed Description 2

## 2.1 General, applications

**General**

The “Software cams” function (see Chapter 3) generates position-dependent switching signals for axes that supply an actual position value (machine axes) and for simulated axes. The cam signals can be output to the PLC as well as 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 MMC, 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.

**Applications**

Examples of cam signal applications are as follows:

- 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 $\geq$ plus cam |
| Cam range minus: | All positions $\leq$ minus cam |

# 2.2 Cam signals and cam positions

## 2.2.1 Generation of cam signals for separate output

**General**

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.

**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.

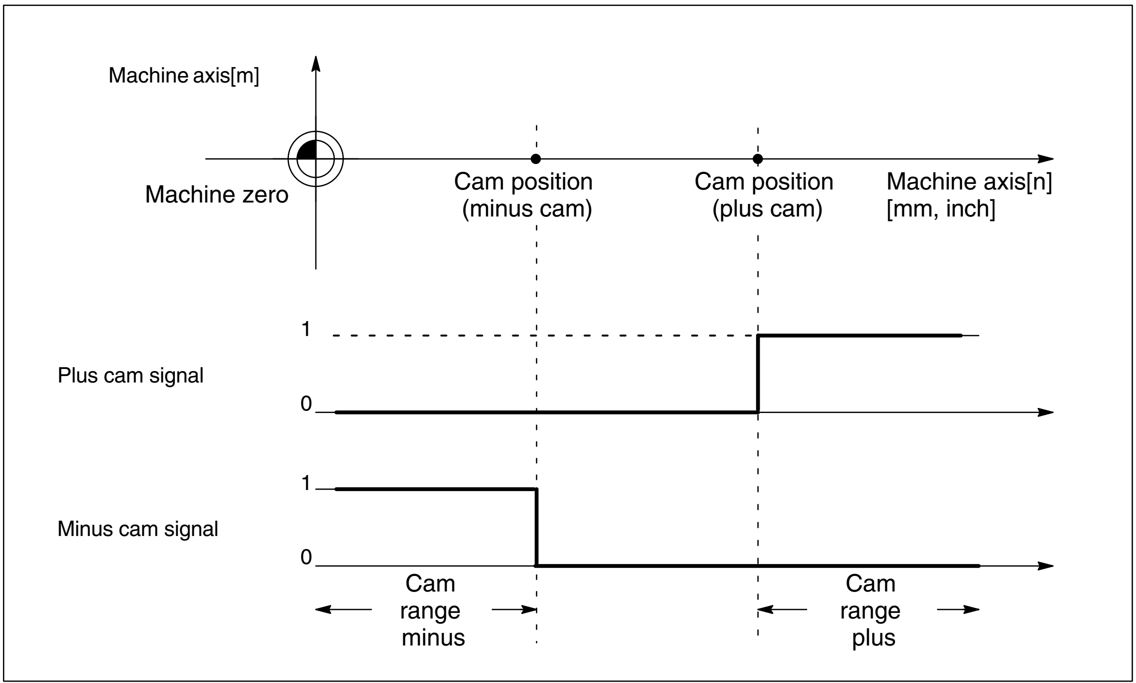



Fig. 2-1 Software cams for linear axis (minus cam < plus cam)

---

**Note**

Position switching signals:

If the axis is positioned exactly on the cam, 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.

---

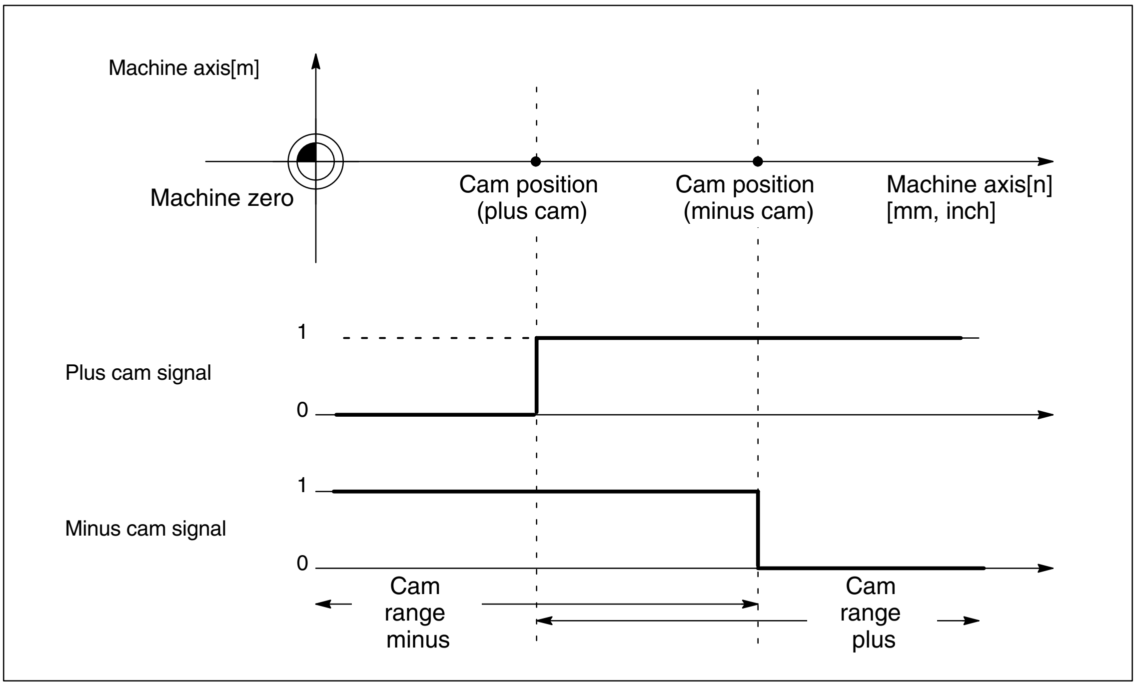



Fig. 2-2 Software cams for linear axis (plus cam < minus cam)

## 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 plus cam response applies under the following **condition**:
Plus cam – minus cam < 180 degrees

---

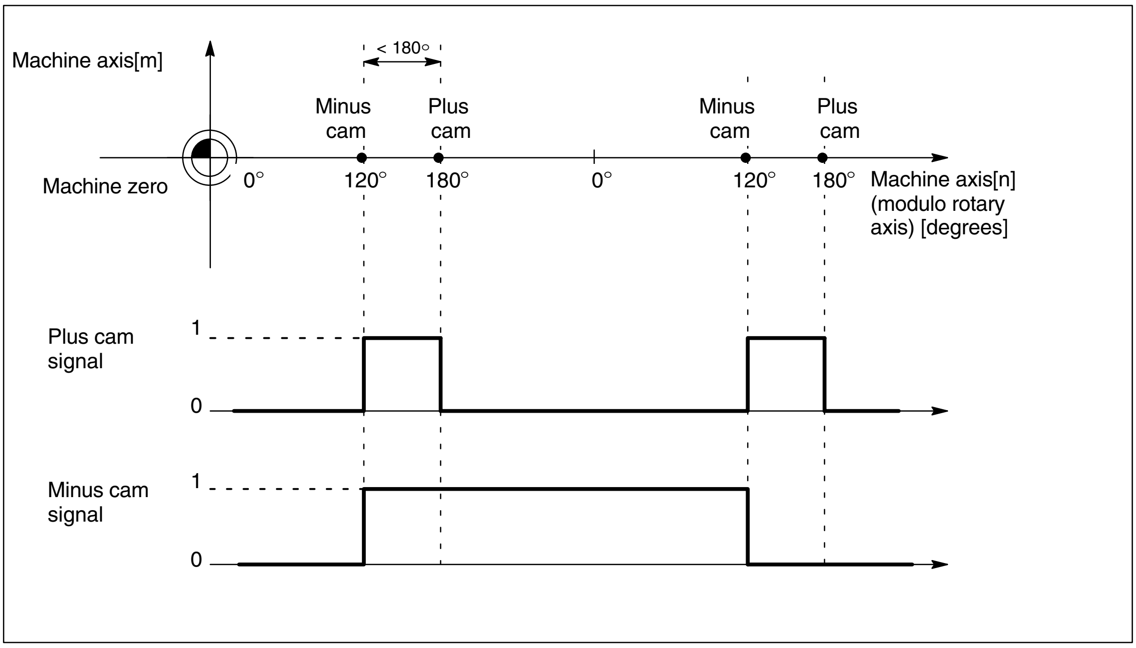



Fig. 2-3 Software cams for modulo rotary axis (plus cam – minus cam < 180 degrees)

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.

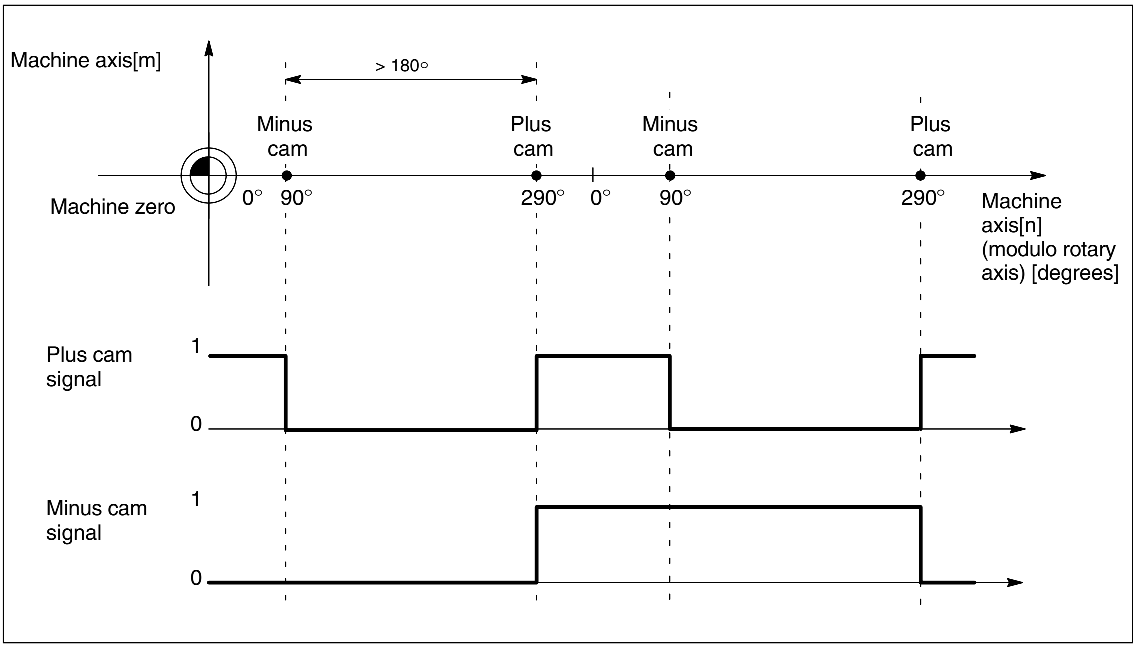



Fig. 2-4 Software cams for modulo rotary axis (plus cam – minus cam > 180 degrees)

## 2.2.2 Generation of cam signals with gated output

**General**

The plus and minus cam output signals are gated in the case of:

- timer-controlled cam signal output (Subsection 2.3.3) to the four onboard outputs on the NCU
- output to the NCK I/O devices if the 2nd byte in MD 10470 to MD 10473 has not been specified (= “0”)

**Linear axes**

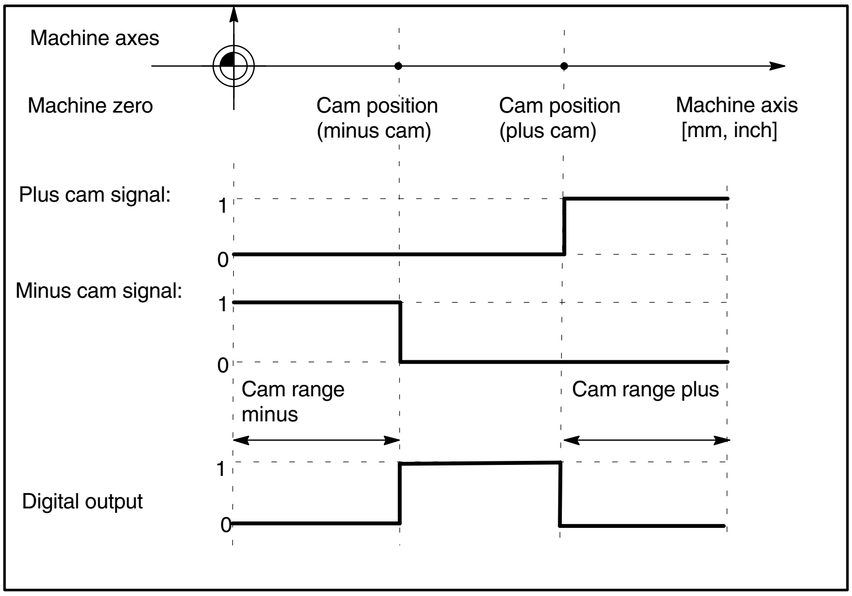



Fig. 2-5 Position switching signals for linear axis (minus cam < plus cam)

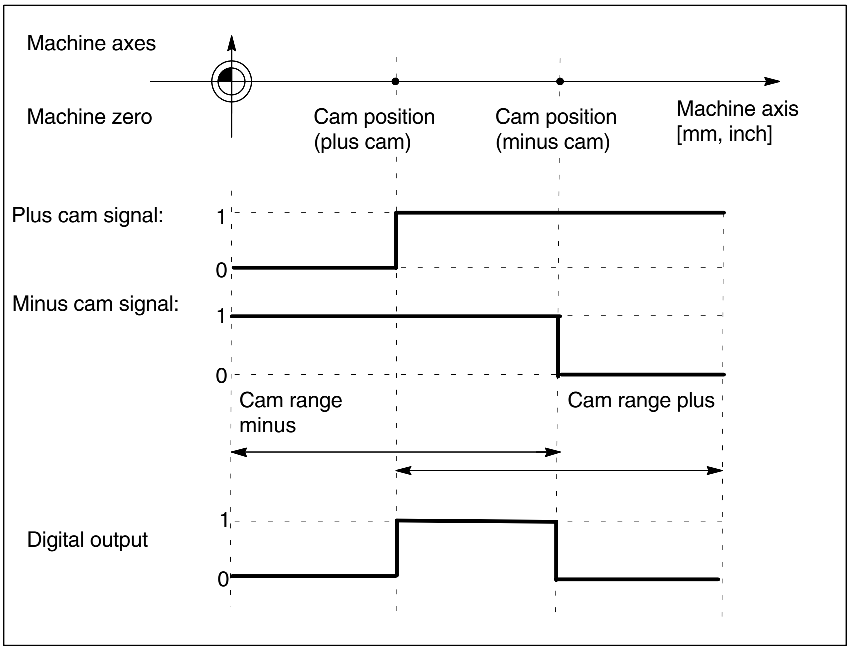



Fig. 2-6 Position switching signals for linear axis (plus cam < minus cam)

**Modulo rotary axis**

The default signal response for modulo rotary axes is dependent on the cam width:

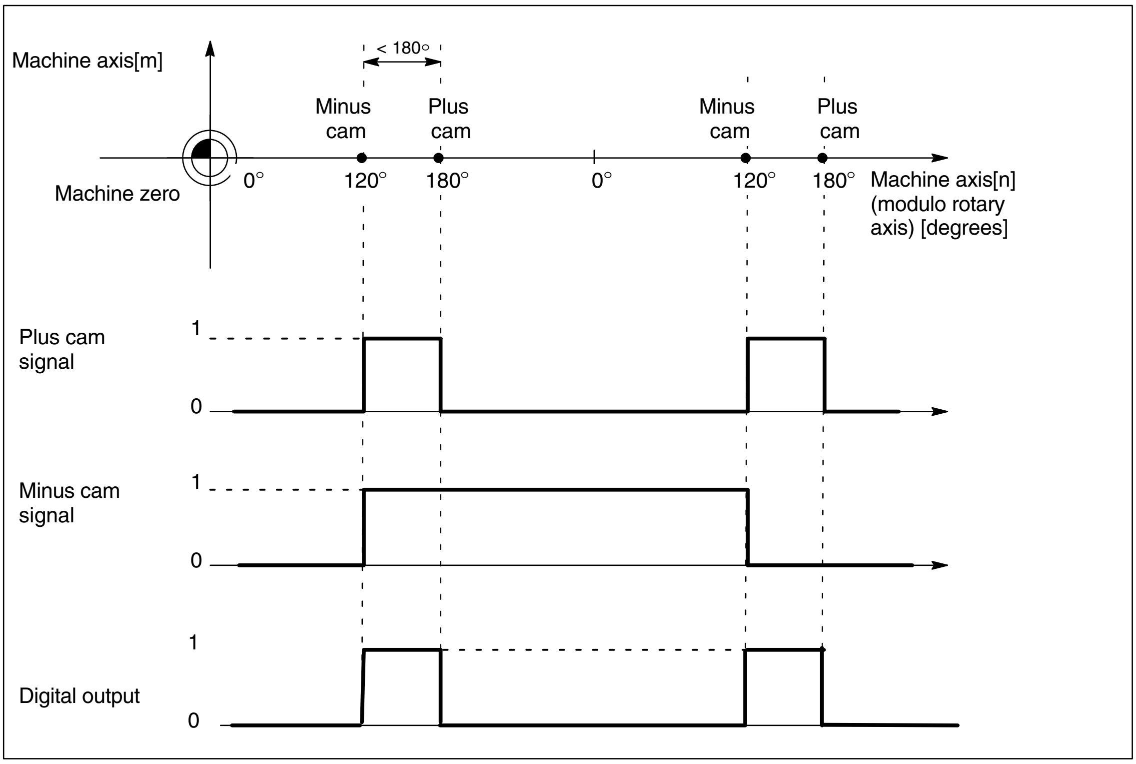



Fig. 2-7 Software cams for modulo rotary axis (plus cam – minus cam < 180 degrees)

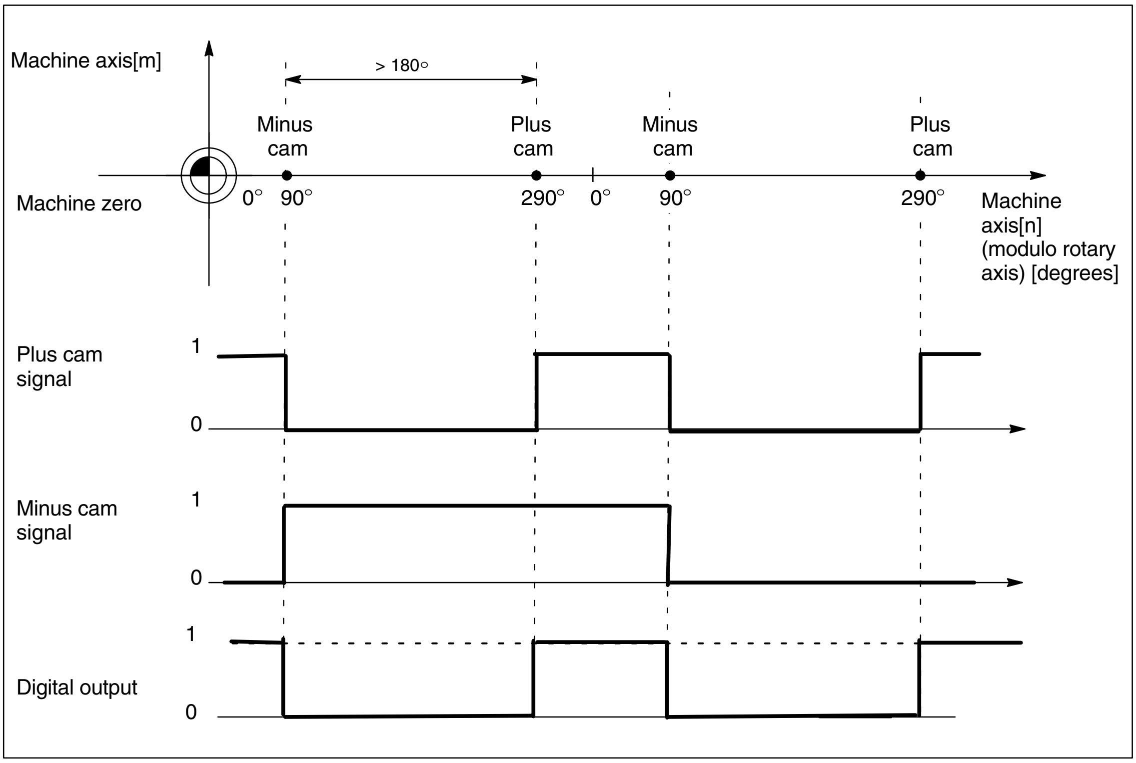



Fig. 2-8 Software cams for modulo rotary axis (plus cam – minus cam > 180 degrees)

**Suppression of signal inversion**

In SW version 6.3 and higher, MD 10485: SW_CAM_MODE Bit 1=1 can be set to suppress the signal inversion for plus cam – minus cam > 180 degrees.

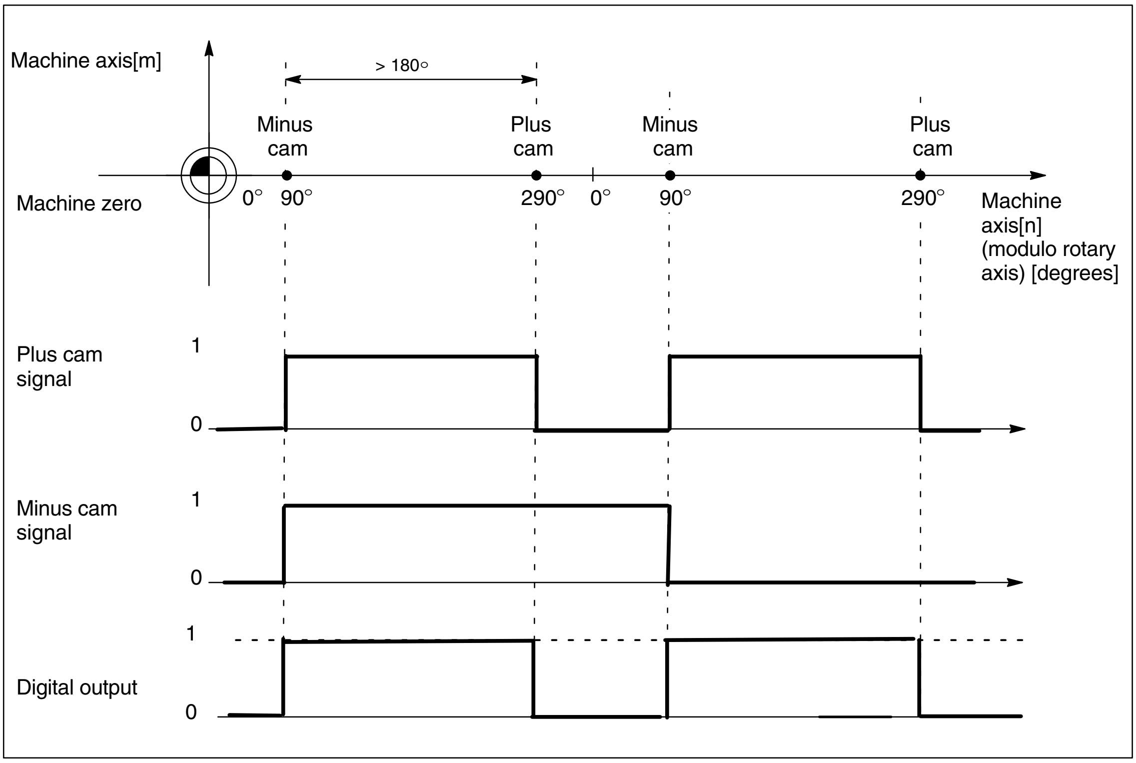



Fig. 2-9 Software cams for modulo rotary axis (plus cam – minus cam > 180 degrees) and suppression of signal inversion

### 2.2.3 Cam positions

**Setting cam positions**

The cam positions of the plus and minus cams are defined via the following general setting data:

SD 41500: SW_CAM_MINUS_POS_TAB_1[n] Position of minus cams 1 – 8
SD 41501: SW_CAM_PLUS_POS_TAB_1[n] Position of plus cams 1 – 8

SD 41502: SW_CAM_MINUS_POS_TAB_2[n] Position of minus cams 9 –16
SD 41503: SW_CAM_PLUS_POS_TAB_2[n] Position of plus cams 9 – 16

In addition, from SW 4.1:

SD 41504: SW_CAM_MINUS_POS_TAB_3[n] Position of minus cams 17 –24
SD 41505: SW_CAM_PLUS_POS_TAB_3[n] Position of plus cams 17 – 24

SD 41506: SW_CAM_MINUS_POS_TAB_4[n] Position of minus cams 25 – 32
SD 41507: 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.

---

**System of units metric/inch**

**From SW 5** and MD 10260: CONVERT_SCALING_SYSTEM=1 (see /G2/) the cam positions no longer refer to the basic system that is set but to the measuring system set in MD 10270: POS_TAB_SCALING_SYSTEM.

MD 10270: POS_TAB_SCALING_SYSTEM=0: metric
MD 10270: POS_TAB_SCALING_SYSTEM=1: inch

The MD 10270 thus defines the measuring system for position specifications from SD 41500 to SD 41507.

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 accessed for reading and writing via the MMC, PLC and part program.
Access operations from the part program are not synchronized with block processing. 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 FB 2 and FB 3 in the PLC user program.

**Axis/cam assignment**

An assignment between a cam pair and a machine axis is made via the general MD 10450: 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.

---

### 2.2.4 Lead/delay times (dynamic cam)

**Times**

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 output of cam signals.

---

**Input in machine data**

The **first** lead or delay time is entered in the following general machine data:

MD 10460: SW_CAM_MINUS_LEAD_TIME[n]
lead or delay time on minus cams
MD 10461: SW_CAM_PLUS_LEAD_TIME[n]
lead or delay time on plus cams

For example, the following entries can be made in these machine data:

- Constant internal delay times between actual-value sensing and cam signal output (e.g. as determined by an oscilloscope) or
- Constant external delay times.

**Input in setting data**

The **second** lead or delay time is entered in the following general setting data:

SD 41520: SW_CAM_MINUS_TIME_TAB_1[n]
lead or delay time on minus cams 1 – 8
SD 41521: SW_CAM_PLUS_TIME_TAB_1[n]
lead or delay time on plus cams 1 – 8

SD 41522: SW_CAM_MINUS_TIME_TAB_2[n]
lead or delay time on minus cams 9 – 16
SD 41523: SW_CAM_PLUS_TIME_TAB_2[n]
lead or delay time on plus cams 9 – 16

SD 41524: SW_CAM_MINUS_TIME_TAB_3[n]
lead or delay time on minus cams 17 –24
SD 41525: SW_CAM_PLUS_TIME_TAB_3[n]
lead or delay time on plus cams 17 – 24

SD 41526: SW_CAM_MINUS_TIME_TAB_4[n]
lead or delay time on minus cams 25 – 32
SD 41527: SW_CAM_PLUS_TIME_TAB_4[n]
lead or delay time on plus cams 25 – 32

Delay times which may **change** during machining must, for example, be entered in these setting data.

## 2.3 Output of cam signals

The cam status, i.e. 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 for an axis is activated via axis-specific **IS "Cam activation"(DB31, ... ; DBX2.0)** .

**Check-back signal to PLC**

Successful activation of all cams for an axis is signaled back to the PLC via axis-specific **IS "Cams active"(DB31, ... ; DBX62.0)** .

---

**Note**

Activation can also be linked to other conditions (e.g. axis referenced, RESET effective) by the PLC user.

---

### 2.3.1 Output of cam signals 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 in the general **IS "Minus cam signal 1 to 32"(DB10, DBX110.0 to 113.7)** .

**Plus cam signals**

The status of the plus cam signals is entered in the general **IS "Plus cam signal 1 to 32"(DB10, DBX114.0 to 117.7)** .

---

**Note**

| If there is ... | then ... |
|---|---|
| no measuring system is selected<br>**or**<br>IS "Cam activation" (DB31, ... ; DBX2.0) = "0" | the **following IS are set to "0"** :<br>–Minus cam signals 1–32 (DB10, DBX110.0–113.7)<br>–Plus cam signals 1–32 (DB10, DBX114.0–117.7)<br>–Cam active (DB31, ... ; DBX62.0) |

---

### 2.3.2 Output of cam signals to NCK I/Os in position control cycle

The signals for the cams assigned via MD 10470 to MD 10473 to a HW byte are output in the position control cycle. Details about switching accuracy are given further below.

The 4 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.

**References:** /FB/, A4, “Digital and Analog NCK I/Os”

**Hardware assignment**

The assignment to the hardware bytes used is made for each eight cam pairs in both the common machine data items

MD 10470: SW_CAM_ASSIGN_FASTOUT_1 Hardware assignment for output of cams 1 – 8 to NCK I/Os

MD 10471: SW_CAM_ASSIGN_FASTOUT_2 Hardware assignment for output of cams 9 – 16 to NCK I/Os

MD 10472: SW_CAM_ASSIGN_FASTOUT_3 Hardware assignment for output of cams 17 – 24 to NCK I/Os

MD 10473: 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. The signal generation for the logic operation is shown in Figs. 2-5 to 2-9.

---

**Status query in the parts program**

The status of the HW outputs can be read in the parts 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. Owing to the time grid of the position control cycle, the switching accuracy of the cam signals is limited as a function of velocity.

In this case: Delta pos = $V_{act}$ * position control cycle

Parameters

Delta pos: Switching accuracy (governed by position control cycle)
$V_{act}$: Current axis velocity

Example

$V_{act}$ = 20 m/min, pos. contr. cycle = 4 ms — Delta pos = 1.33 mm
$V_{act}$ = 2000 rev/min, pos. contr. cycle = 2 ms — Delta pos = 24 degrees

### 2.3.3 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.

General MD 10480: SW_CAM_TIMER_FASTOUT_MASK (mask for the output of cam signals via timer interrupts on NCU) can be set to select timer-controlled output to the 4 NCU onboard outputs for 4 cam pairs.

In this case, the minus and plus signals of a cam pair are gated for output as one signal. The signal generation for the logic operation is shown in Figs. 2-5 to 2-9.

**Signal generation**

MD 10485: SW_CAM_MODE bit 1 must first be set to define the method of generation of the signals to be gated. If the bit is not set, the signals are generated according to 2.2.1:

Inversion of signal response of plus cam when
plus cam – minus cam >= 180 degrees

If the bit is set, then the signal response shown in 2.2.2 is selected:

**No** inversion of signal response of plus cam when
plus cam – minus cam >= 180 degrees

The setting option is available in SW version 6.2 and higher. In earlier software versions without MD 10485: SW_CAM_MODE the variant with inversion shown in 2.2.1 is always set.

---

**Note**

This function works independently of the assignment set in
MD 10470: SW_CAM_ASSIGN_FASTOUT_1 or
MD 10471: SW_CAM_ASSIGN_FASTOUT_2 or
MD 10472: SW_CAM_ASSIGN_FASTOUT_3 or
MD 10473: SW_CAM_ASSIGN_FASTOUT_4.

The onboard byte may not be used more than once at any one time.

---

**Restrictions**

The following applies to the mutual position of the cam positions:

Only **one** signal is output on a timer-controlled basis per IPO cycle. If there are signal changes for more than one cam pair in an IPO cycle, then the signals are output on a priority basis:
The cam pair with the lowest number (1...32) determines the instant at which **all** pending signals are output, i.e. the signal change of the other cam pairs takes place at the same instant in 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.

These signals are irrelevant, however, or correspondingly inaccurate with the **timer-controlled** cam output variant, as described in the following paragraphs.
The signals for plus and minus cams are generated synchronously (once) in the interpolation cycle and passed together to the PLC.
Pulses shorter than one interpolation cycle are thus not visible in the PLC. The onboard outputs are set and reset by interrupt asynchronously to the interpolation cycle.
The status of the onboard outputs are acquired in synchronism with the update of the PLC interface and transferred to the PLC.
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**

General MD 10485: SW_CAM_MODE is available in SW version 6.2 and higher. If the signal response described here is to be selected, then bit 0 must be set to 0 in this MD. This MD does not exist in older SW versions. The described response is implicitly selected if MD 10480: SW_CAM_FASTOUT_MASK is set accordingly.

### 2.3.4 Independent, timer-controlled output of cam signals (SW 6.2 and higher)

Each switching edge is output separately per interrupt due to the timer-controlled, independent (of interpolation cycle) cam output. The interaction between cam signals described in 2.3.3 as a result of:

- single output per interpolation cycle
- output time determined by highest priority cam pair (lowest cam pair number)

does not take place.

**A total of eight 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, but they are not relevant or accurate with a timer-controlled output.

**Signal generation**

MD 10485: SW_CAM_MODE bit 1 must first be set to define the method of generation of the signals to be gated. The signal generation for the logic operation is shown in Figs. 2-5 to 2-9. If the bit is not set, the signals are generated according to 2.2.1:

Inversion of signal response of plus cam when
plus cam – minus cam >= 180 degrees

If the bit is set, then the signal response shown in 2.2.2 is selected:

**No** inversion of signal response of plus cam when
plus cam – minus cam >= 180 degrees

**Settings**

The assignment of cams pair to onboard outputs is parameterized, as with 2.3.3, in MD 10480: SW_CAM_TIMER_FASTOUT_MASK (screen form for output of cam signals via timer interrupts on NCU). This processing method must also be selected explicitly via general MD 10485: SW_CAM_MODE bit 0=1.

---

**Note**

This function works independently of the assignment set in
MD 10470: SW_CAM_ASSIGN_FASTOUT_1 or
MD 10471: SW_CAM_ASSIGN_FASTOUT_2 or
MD 10472: SW_CAM_ASSIGN_FASTOUT_3 or
MD 10473: SW_CAM_ASSIGN_FASTOUT_4.

The onboard byte may not be used more than once at any one time.

---

## 2.4 Position-time cam

**Definition**

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. MD can be parameterized to select that cam pairs with
minus cam position = plus cam position
must 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).
- The cam is activated (ON edge) only when the cam position is crossed. Moving the axis position (e.g. Preset) does not activate the cam.
- 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 = the plus cam position. The position is parameterized according to 2.2.3 by setting data 41500 to 41507.

**Pulse duration:**
The pulse duration is the product of the associated entries for the cam pair in:
MD10461: SW_CAM_PLUS_LEAD_TIME[n] and
SD 41521: SW_CAM_PLUS_TIME_TAB_1[n]...
SD 41527: SW_CAM_PLUS_TIME_TAB_4[n]

**Displacement:**
The time displacement of the position-time cam is the product of the associated entries for the cam pair in:
MD10460: SW_CAM_MINUS_LEAD_TIME[n] and
SD 41520: SW_CAM_MINUS_TIME_TAB_1[n]...
SD 41526: SW_CAM_MINUS_TIME_TAB_4[n]

**Mode:**
Bit 2 = 1 must be set in MD 10485: to ensure that all cam pairs with the same values for minus cam and plus cam positions are treated as position-time cams.

■

## Notes

# Supplementary Conditions 3

**Availability of function “Software cams, position switching signals”**

The function is an option and is available for

- SINUMERIK 840D with NCU 572/573, SW2 and higher
- SINUMERIK 810D, SW 3.2 and higher

**Extensions**

The extension:

- 32 instead of 16 cam pairs

is available in SW 4.1and higher.

The extensions:

- Independent, timer-controlled output (see 2.3.4)
- Position-time cam (see 2.4)
- Suppression of signal inversion for timer-controlled outputs (see 2.2.2)

are available in SW 6.3 and higher.

■

# Notes

# Data Descriptions (MD, SD) 4

## 4.1 General machine data

| | | | |
|---|---|---|---|
| **10450**<br>MD number | **SW_CAM_ASSIGN_TAB[n]**<br>Assignment of software cams to machine axes | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 8 or 31 | |
| Changes effective after POWER ON | Protection level: 2/4 | Unit: – | |
| Data type: BYTE | Applies from SW: 2.1 or 4.1 | | |
| Meaning: | This machine data allows one machine axis to be assigned to each of the 32 (16) possible cam pairs (comprising one minus and one plus cam).<br>When a “0” is entered, the appropriate cam is not processed.<br>The cam signal output is activated via the axial IS “Cam activation” (DB31, ... ;, DBX2.0).<br>Index [n] of the machine data addresses the cam pair:<br>n = 0, 1, ... , 31 corresponds to cam pair 1, 2, ... , 32 | | |
| Application example(s) | Cam pair 1 must be assigned to machine axis 3 and cam pair 3 to machine axis 4.<br>Cam pair 2 is not to be assigned to any axis.<br>⇒MD: SW_CAM_ASSIGN_TAB[0]= 3<br>MD: SW_CAM_ASSIGN_TAB[1]= 0<br>MD: SW_CAM_ASSIGN_TAB[2]= 4 | | |
| Related to .... | IS “Cam activation” (DB31-48, DBX2.0) | | |

| | | | | |
|---|---|---|---|---|
| **10460**<br>MD number | **SW_CAM_MINUS_LEAD_TIME[n]**<br>Lead or delay time on minus cams 1 – 16 | | | |
| Default setting: 0.0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective after POWER ON | | Protection level: 2/4 | | Unit: s |
| Data type: DOUBLE | | | Applies from SW: 2.1 | |
| Meaning: | A lead or delay time can be assigned to each **minus cam 1–16** in this machine data to compensate for delay times.<br><br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br><br>Positive value: ⇒ Lead time<br>Negative value: ⇒ Delay time<br><br>Index [n] of the machine data addresses the cam pair:<br>n = 0, 1, ... , 15 corresponds to cam pair 1, 2, ... , 16<br><br>This machine data is added to setting data<br>SW_CAM_MINUS_TIME_TAB_1[n] and SW_CAM_MINUS_TIME_TAB_2[n]. | | | |
| Related to .... | SD: SW_CAM_MINUS_TIME_TAB_1[n] (lead or delay time on minus cams 1 – 8)<br><br>SD: SW_CAM_MINUS_TIME_TAB_2[n] (lead or delay time on minus cams 9 – 16) | | | |

| | | | | |
|---|---|---|---|---|
| **10461**<br>MD number | **SW_CAM_PLUS_LEAD_TIME[n]**<br>Lead or delay time on plus cams 1 – 16 | | | |
| Default setting: 0.0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective after POWER ON | | Protection level: 2/4 | | Unit: s |
| Data type: DOUBLE | | | Applies from SW: 2.1 | |
| Meaning: | A lead or delay time can be assigned to each **plus cam 1–16** in this machine data to compensate for delay times.<br><br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br><br>Positive value: ⇒ Lead time<br>Negative value: ⇒ Delay time<br><br>Index [n] of the machine data addresses the cam pair:<br>n = 0, 1, ... , 15 corresponds to cam pair 1, 2, ... , 16<br><br>This machine data is added to setting data SW_CAM_PLUS_TIME_TAB_1[n] and SW_CAM_PLUS_TIME_TAB_2[n]. | | | |
| Related to .... | SD: SW_CAM_PLUS_TIME_TAB_1[n] (lead or delay time on plus cams 1 – 8)<br><br>SD: SW_CAM_PLUS_TIME_TAB_2[n] (lead or delay time on plus cams 9 – 16) | | | |

| | | |
|---|---|---|
| **10470**<br>MD number | **SW_CAM_ASSIGN_FASTOUT_1**<br>Hardware assignment for output of cams 1 – 8 to NCK I/Os | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective after POWER ON | Protection level: 2/4 | Unit: HEX |
| Data type: DWORD | Applies from SW: 2.1 | |
| Meaning: | The cam signal status can be output to the NCK I/Os as well as to the PLC.<br><br>The hardware assignment of the minus and plus cam signals to the digital output bytes used can be made in this machine data for **cam pairs 1 –8** .<br><br>In addition, the assigned output signals can be inverted with this machine data.<br><br>The MD is coded as follows:<br>Bit 0–7: Number of 1st HW byte used with digital outputs<br>Bit 8–15: Number of 2nd HW byte used with digital outputs<br><br>Bit 16–23: Inversion screen form for writing 1st HW byte used<br>Bit 24–31: Inversion screen form for writing 2nd HW byte used<br>Bit=0: Do not invert<br>Bit=1: Invert<br><br>If both HW bytes are specified, the 1st byte contains the minus cam signals and the 2nd byte the plus cam signals.<br><br>If the 2nd byte is not specified (= “0”), then the 8 cams are output as an AND operation of the minus and plus cam signals via the 1st HW byte using the 1st inversion screen form. The status of the non-inverted output signal for linear axes and for rotary axes with “plus cam – minus cam < 180 degrees is:<br>“1” between minus and plus cams<br>“0” outside this range<br><br>The status of the non-inverted output signal for rotary axes with “plus cam – minus cam ≥ 180 degrees”:<br>“0” between minus and plus cams<br>“1” outside this range<br><br>The following must be specified as the byte address for the digital outputs:<br>1: for onboard byte<br>2 – 5: for external bytes | |
| Application example(s) | The minus cam signals must be output via the onboard byte.<br><br>The plus cam signals must be output via byte address 3 on the NCU terminal block.<br><br>The following must also be inverted:<br>Minus cam signal 2, 4, 5 (corresponds to bits 1, 3, 4 of 1st HW byte)<br>Plus cam signal 1, 3, 4 (corresponds to bits 0, 2, 3 or 2nd HW byte)<br><br>⇒MD: SW_CAM_ASSIGN_FASTOUT_1=‘H0D1A0301’ | |

| | | | | | |
|---|---|---|---|---|---|
| **10471**<br>MD number | **SW_CAM_ASSIGN_FASTOUT_2**<br>Hardware assignment for output of cams 9 – 16 to NCK I/Os | | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** | |
| Changes effective after POWER ON | | | Protection level: 2/4 | | Unit: HEX |
| Data type: DWORD | | | | Applies from SW: 2.1 | |
| Meaning: | The cam signal status can be output to the NCK I/Os as well as to the PLC.<br><br>The hardware assignment of the minus and plus cam signals to the digital output bytes used can be made in this machine data for **cam pairs 9 –16** .<br><br>In addition, the assigned output signals can be inverted with this machine data.<br><br>The MD is coded as follows:<br>Bit 0–7: Number of 1st HW byte used with digital outputs<br>Bit 8–15: Number of 2nd HW byte used with digital outputs<br><br>Bit 16–23: Inversion screen form for writing 1st HW byte used<br>Bit 24–31: Inversion screen form for writing 2nd HW byte used<br>Bit=0: Do not invert<br>Bit=1: Invert<br><br>If both HW bytes are specified, the 1st byte contains the minus cam signals and the 2nd byte the plus cam signals.<br><br>If the 2nd byte is not specified (= “0”), then the 8 cams are output as an AND operation of the minus and plus cam signals via the 1st HW byte using the 1st inversion screen form. The status of the non-inverted output signal for linear axes and for rotary axes with “plus cam – minus cam < 180 degrees is:<br>“1” between minus and plus cams<br>“0” outside this range<br><br>The status of the non-inverted output signal for rotary axes with “plus cam – minus cam ≥ 180 degrees”:<br>“0” between minus and plus cams<br>“1” outside this range<br><br>The following must be specified as the byte address for the digital outputs:<br>1: for onboard byte<br>2 – 5: for external bytes | | | | |
| Application example(s) | See MD: SW_CAM_ASSIGN_FASTOUT_1 | | | | |

<table>
<tr><td>10472<br>MD number</td><td colspan="3">SW_CAM_ASSIGN_FASTOUT_3<br>Hardware assignment for output of cams 17 – 24 to NCK I/Os</td></tr>
<tr><td>Default setting: 0</td><td>Minimum input limit: ***</td><td colspan="2">Maximum input limit: ***</td></tr>
<tr><td>Changes effective after POWER ON</td><td>Protection level: 2/7</td><td colspan="2">Unit: HEX</td></tr>
<tr><td>Data type: DWORD</td><td colspan="3">Applies from SW: 4.1</td></tr>
<tr><td>Meaning:</td><td colspan="3">The cam signal status can be output to the NCK I/Os as well as to the PLC.<br><br>The hardware assignment of the minus and plus cam signals to the digital output bytes used can be made in this machine data for cam pairs 17 –24 .<br><br>In addition, the assigned output signals can be inverted with this machine data.<br><br>The MD is coded as follows:<br>Bit 0–7: Number of 1st HW byte used with digital outputs<br>Bit 8–15: Number of 2nd HW byte used with digital outputs<br><br>Bit 16–23: Inversion screen form for writing 1st HW byte used<br>Bit 24–31: Inversion screen form for writing 2nd HW byte used<br>Bit=0: Do not invert<br>Bit=1: Invert<br><br>If both HW bytes are specified, the 1st byte contains the minus cam signals and the 2nd byte the plus cam signals.<br><br>If the 2nd byte is not specified (= “0”), then the 8 cams are output as an AND operation of the minus and plus cam signals via the 1st HW byte using the 1st inversion screen form.<br>The status of the non-inverted output signal for linear axes and for rotary axes with “plus cam – minus cam < 180 degrees is:<br>“1” between minus and plus cams<br>“0” outside this range<br><br>The status of the non-inverted output signal for rotary axes with “plus cam – minus cam ≥ 180 degrees”:<br>“0” between minus and plus cams<br>“1” outside this range<br><br>The following must be specified as the byte address for the digital outputs:<br>1: for onboard byte<br>2 – 5: for external bytes</td></tr>
<tr><td>Application example(s)</td><td colspan="3">See MD: SW_CAM_ASSIGN_FASTOUT_1</td></tr>
</table>

| | |
|---|---|
| **10473**<br>MD number | **SW_CAM_ASSIGN_FASTOUT_4**<br>Hardware assignment for output of cams 25 – 32 to NCK I/Os |
| Default setting: 0 | Minimum input limit: *** Maximum input limit: *** |
| Changes effective after POWER ON | Protection level: 2/7 Unit: HEX |
| Data type: DWORD | Applies from SW: 4.1 |
| Meaning: | The cam signal status can be output to the NCK I/Os as well as to the PLC.<br><br>The hardware assignment of the minus and plus cam signals to the digital output bytes used can be made in this machine data for **cam pairs 25 –32** .<br><br>In addition, the assigned output signals can be inverted with this machine data.<br><br>The MD is coded as follows:<br>Bit 0–7: Number of 1st HW byte used with digital outputs<br>Bit 8–15: Number of 2nd HW byte used with digital outputs<br><br>Bit 16–23: Inversion screen form for writing 1st HW byte used<br>Bit 24–31: Inversion screen form for writing 2nd HW byte used<br>Bit=0: Do not invert<br>Bit=1: Invert<br><br>If both HW bytes are specified, the 1st byte contains the minus cam signals and the 2nd byte the plus cam signals.<br><br>If the 2nd byte is not specified (= “0”), then the 8 cams are output as an AND operation of the minus and plus cam signals via the 1st HW byte using the 1st inversion screen form. The status of the non-inverted output signal for linear axes and for rotary axes with “plus cam – minus cam < 180 degrees is:<br>“1” between minus and plus cams<br>“0” outside this range<br><br>The status of the non-inverted output signal for rotary axes with “plus cam – minus cam ≥ 180 degrees”:<br>“0” between minus and plus cams<br>“1” outside this range<br><br>The following must be specified as the byte address for the digital outputs:<br>1: for onboard byte<br>2 – 5: for external bytes |
| Application example(s) | See MD: SW_CAM_ASSIGN_FASTOUT_1 |

<table>
<tr><td>10480<br>MD number</td><td colspan="4">SW_CAM_TIMER_FASTOUT_MASK<br>Screen form for output of cam signals via timer interrupts to NCU</td></tr>
<tr><td colspan="2">Default setting: 0</td><td colspan="2">Minimum input limit: ***</td><td>Maximum input limit: ***</td></tr>
<tr><td colspan="2">Changes effective after POWER ON</td><td colspan="2">Protection level: 2/4</td><td>Unit: HEX</td></tr>
<tr><td colspan="3">Data type: DWORD</td><td colspan="2">Applies from SW: 2.1</td></tr>
<tr><td>Meaning:</td><td colspan="4">A timer-controlled output to the 4 onboard outputs of the NCK I/Os can be selected in this machine data for four cam pairs.<br><br>In this case, the minus and plus signals of a cam pair are “EXCLUSIVE ORed” for output as one signal.<br><br>Meaning for set bit:<br>Associated cam (minus and plus cam signals EXCLUSIVE ORed) is output via a timer interrupt at one of the four onboard outputs of the NCU.<br><br>The onboard outputs are allocated in ascending order of machine axis numbers (with assigned cam pairs).<br><br>Example 1:<br><br>Machine axis 3 = Cam pair 1 —> Onboard output 3<br>Machine axis 1 = Cam pair 2 —> Onboard output 1<br>Machine axis 7 = Cam pair 3 —> Onboard output 4<br>Machine axis 2 = Cam pair 4 —> Onboard output 2<br><br>If several cam pairs are set for a machine axis, the allocation for this axis is in ascending order of cam pairs.<br><br>Example 2:<br><br>Machine axis 3 = Cam pair 1 —> Onboard output 2<br>Machine axis 3 = Cam pair 2 —> Onboard output 3<br>Machine axis 7 = Cam pair 3 —> Onboard output 4<br>Machine axis 2 = Cam pair 4 —> Onboard output 1<br><br>This function works independently of the assignment set in MD: SW_CAM_ASSIGN_FASTOUT_1 or MD: SW_CAM_ASSIGN_FASTOUT_2.<br><br>Note: The onboard byte may not be used more than once at any one time.<br><br>If there is more than one signal change in the IPO cycle for the cam pairs specified in the MD, then the cam pair with the lowest number determines the instant of output. The other signal changes take place at the same time.<br><br>SW 6.3 and higher If MD 10485 bit 0 = 1, independent, timer-controlled output, up to 8 cam signals per interpolation cycle can be output differentially to the onboard outputs. (see 2.3.4)</td></tr>
<tr><td>Application example(s)</td><td colspan="4">The signals of cam pairs 2, 5 and 7 must be output on a timer-controlled basis:<br><br>⇒ MD: SW_CAM_TIMER_FASTOUT_MASK= ‘H52’<br><br>⇒ Signal for cam pair 2 to onboard output 1 of NCK<br>Signal for cam pair 5 to onboard output 2 of NCK<br>Signal for cam pair 7 to onboard output 3 of NCK</td></tr>
<tr><td>Related to ....</td><td colspan="4">MD 10485: SW_CAM_MODE</td></tr>
</table>

<table>
<tr><td>10485<br>MD number</td><td colspan="4">SW_CAM_MODE<br>Response of SW cams</td></tr>
<tr><td colspan="2">Default setting: 0</td><td colspan="2">Minimum input limit: 0</td><td>Maximum input limit: 7</td></tr>
<tr><td colspan="2">Changes effective after POWER ON</td><td colspan="2">Protection level: 2 / 7</td><td>Unit: –</td></tr>
<tr><td colspan="3">Data type: DWORD</td><td colspan="2">Applies from SW: 6.3</td></tr>
</table>

| **10485**<br>MD number | **SW_CAM_MODE**<br>Response of SW cams |
|---|---|
| Meaning: | Definition of response of SW cams:<br><br>Bit 0:(LSB) Independent timer-controlled cam output<br>Bit 1: Suppression of signal inversion for timer-controlled cam output<br>Bit 2: Position-time cam<br>Bit 3: Reserved<br>Bit 4: Unassigned<br>Meaning of individual bits:<br>Bit 0 = 0: If there is more than 1 signal change in the IPO cycle for the cam specified in MD SW_CAM_TIMER_FAST OUT_MASK, then the cam with the lowest number determines the time of output. The other signal changes take place at the same time, i.e. a maximum of **one** interrupt-driven output takes place in each interpolation cycle.<br>Bit 0 = 1: Every cam specified in MD SW_CAM_TIMER_ FASTOUT_MASK is output at exactly the right time in the interpolation cycle. No output priorities apply to the cams. A maximum of **eight** interrupt-driven outputs take place in each interpolation cycle.<br><br>Bit 1 = 0: Inversion of the signal response of the plus cam when plus cam – minus cam >= 180 deg.<br>Bit 1 = 1: **No** inversion of signal response of plus cam when plus cam – minus cam >= 180 deg.<br><br>**Signal response onboard output:**<br>Overtraveling:<br><br>Minus cam / Plus cam<br>Travel direction:<br>Positive: 0–>1 / 1–>0<br>Negative: 1–>0 / 0–>1<br><br>Bit 2 = 0: Not a position-time cam<br>Bit 2 = 1: Position-time cam for cams with minus position = plus position<br>The applied lead/delay time (plus cam) is operative irrespective of:<br>– the axis velocity<br>– the axis position<br>– a reversal in the direction of travel<br><br>The cam is activated only when the cam position is crossed.<br>A lead/delay time for the minus cam is effective and results in displacement of the entire cam<br><br>Bit 3 Reserved<br><br>Bit 4 = 0: and following unassigned |
| | |
| Related to .... | MD 10480: SW_CAM_TIMER_FASTOUT_MASK |
| References | FM 357-2 multi-axis module |

# 4.2 General setting data

| **41500** SD number | **SW_CAM_MINUS_POS_TAB_1[n]** Position of minus cams 1 – 8 | |
|---|---|---|
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective immediately | Protection level: 7/7 | Unit: mm, degrees |
| Data type: DOUBLE | Applies from SW: 2.1 | |
| Meaning: | The cam position of **minus cams 1 –8** is entered in this machine data.<br>The positions are entered in the machine coordinate system.<br>The response when the cam positions are overtraveled by linear and modulo rotary axes is described in Subsection 2.2.1.<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 1, 2, ... , 8 | |

| **41501** SD number | **SW_CAM_PLUS_POS_TAB_1[n]** Position of plus cams 1 – 8 | |
|---|---|---|
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective immediately | Protection level: 7/7 | Unit: mm, degrees |
| Data type: DOUBLE | Applies from SW: 2.1 | |
| Meaning: | The cam position of **plus cams 1 –8** is entered in this machine data.<br>The positions are entered in the machine coordinate system.<br>The response when the cam positions are overtraveled by linear and modulo rotary axes is described in Subsection 2.2.1.<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 1, 2, ... , 8 | |

| **41502** SD number | **SW_CAM_MINUS_POS_TAB_2[n]** Position of minus cams 9 – 16 | |
|---|---|---|
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective immediately | Protection level: 7/7 | Unit: mm, degrees |
| Data type: DOUBLE | Applies from SW: 2.1 | |
| Meaning: | The cam position of **minus cams 9 –16** is entered in this machine data.<br>The positions are entered in the machine coordinate system.<br>The response when the cam positions are overtraveled by linear and modulo rotary axes is described in Subsections 2.2.1 and 2.2.2.<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 9, 10, ... , 16 | |

| | | | | |
|---|---|---|---|---|
| **41503**<br>SD number | **SW_CAM_PLUS_POS_TAB_2[n]**<br>Position of plus cams 9 – 16 | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective immediately | | | Protection level: 7/7 | Unit: mm, degrees |
| Data type: DOUBLE | | | Applies from SW: 2.1 | |
| Meaning: | The cam position of **plus cams 9 –16** is entered in this machine data.<br>The positions are entered in the machine coordinate system.<br>The response when the cam positions are overtraveled by linear and modulo rotary axes is described in Subsections 2.2.1 and 2.2.2.<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 9, 10, ... , 16 | | | |

| | | | | |
|---|---|---|---|---|
| **41504**<br>SD number | **SW_CAM_MINUS_POS_TAB_3[n]**<br>Position of minus cams 17 – 24 | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective immediately | | | Protection level: 7/7 | Unit: mm, degrees |
| Data type: DOUBLE | | | Applies from SW: 4.1 | |
| Meaning: | The cam position of **minus cams 17 –24** is entered in this machine data.<br>The positions are entered in the machine coordinate system.<br>The response when the cam positions are overtraveled by linear and modulo rotary axes is described in Subsections 2.2.1 and 2.2.2.<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 17, 18, ... , 24 | | | |

| | | | | |
|---|---|---|---|---|
| **41505**<br>SD number | **SW_CAM_PLUS_POS_TAB_3[n]**<br>Position of plus cams 17 – 24 | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective immediately | | | Protection level: 7/7 | Unit: mm, degrees |
| Data type: DOUBLE | | | Applies from SW: 4.1 | |
| Meaning: | The cam position of **plus cams 17 –24** is entered in this machine data.<br>The positions are entered in the machine coordinate system.<br>The response when the cam positions are overtraveled by linear and modulo rotary axes is described in Subsections 2.2.1 and 2.2.2.<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 17, 18, ... , 24 | | | |

| | | |
|---|---|---|
| **41506**<br>SD number | **SW_CAM_MINUS_POS_TAB_4[n]**<br>Position of minus cams 25 – 32 | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective immediately | Protection level: 7/7 | Unit: mm, degrees |
| Data type: DOUBLE | Applies from SW: 4.1 | |
| Meaning: | The cam position of **plus cams 25 –32** is entered in this machine data.<br>The positions are entered in the machine coordinate system.<br>The response when the cam positions are overtraveled by linear and modulo rotary axes is described in Subsections 2.2.1 and 2.2.2.<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 25, 26, ... , 32 | |

| | | |
|---|---|---|
| **41507**<br>SD number | **SW_CAM_PLUS_POS_TAB_4[n]**<br>Position of plus cams 25 – 32 | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective immediately | Protection level: 7/7 | Unit: mm, degrees |
| Data type: DOUBLE | Applies from SW: 4.1 | |
| Meaning: | The cam position of **plus cams 25 –32** is entered in this machine data.<br>The positions are entered in the machine coordinate system.<br>The response when the cam positions are overtraveled by linear and modulo rotary axes is described in Subsections 2.2.1 and 2.2.2.<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 25, 26, ... , 32 | |

| | | |
|---|---|---|
| **41520**<br>SD number | **SW_CAM_MINUS_TIME_TAB_1[n]**<br>Lead or delay time on minus cams 1 – 8 | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective immediately | Protection level: 7/7 | Unit: s |
| Data type: DOUBLE | Applies from SW: 2.1 | |
| Meaning: | A lead or delay time can be assigned to each<br>**minus cam 1 -8** in this setting data to compensate for delay times.<br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br>Positive value: Lead time<br>Negative value: Delay time<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 1, 2, ... , 8<br>This setting data is added to MD: SW_CAM_MINUS_LEAD_TIME[n].<br>**SW 6.3 and higher:** The time settings act as a displacement in the trigger position on position-time cams. | |
| Related to .... | MD: SW_CAM_MINUS_LEAD_TIME[n] (lead or delay time on minus cams 1 – 32) | |

| | | | | |
|---|---|---|---|---|
| **41521**<br>SD number | **SW_CAM_PLUS_TIME_TAB_1[n]**<br>Lead or delay time on plus cams 1 – 8 | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective immediately | | | Protection level: 7/7 | Unit: s |
| Data type: DOUBLE | | | Applies from SW: 2.1 | |
| Meaning: | A lead or delay time can be assigned to each **plus cam 1 –8** in this setting data to compensate for delay times.<br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br>Positive value: Lead time<br>Negative value: Delay time<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 1, 2, ... , 8<br>This setting data is added to MD: SW_CAM_PLUS_LEAD_TIME[n].<br>**SW 6.3 and higher:** The time settings act as the pulse duration on position-time cams. | | | |
| Related to .... | MD: SW_CAM_PLUS_LEAD_TIME[n] (lead or delay time on plus cams 1 – 32) | | | |

| | | | | |
|---|---|---|---|---|
| **41522**<br>SD number | **SW_CAM_MINUS_TIME_TAB_2[n]**<br>Lead or delay time on minus cams 9 – 16 | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective immediately | | | Protection level: 7/7 | Unit: s |
| Data type: DOUBLE | | | Applies from SW: 2.1 | |
| Meaning: | A lead or delay time can be assigned to each **minus cam 9 –16** in this setting data to compensate for delay times.<br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br>Positive value: Lead time<br>Negative value: Delay time<br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 9, 10, ... , 16<br>This setting data is added to MD: SW_CAM_MINUS_LEAD_TIME[n+8].<br>**SW 6.3 and higher:** The time settings act as a displacement in the trigger position on position-time cams. | | | |
| Related to .... | MD: SW_CAM_MINUS_LEAD_TIME[n] (lead or delay time on minus cams 1 – 32) | | | |

| | | | | |
|---|---|---|---|---|
| **41523**<br>SD number | **SW_CAM_PLUS_TIME_TAB_2[n]**<br>Lead or delay time on plus cams 9 – 16 | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective immediately | | | Protection level: 7/7 | Unit: s |
| Data type: DOUBLE | | | Applies from SW: 2.1 | |

| **41523**<br>SD number | **SW_CAM_PLUS_TIME_TAB_2[n]**<br>Lead or delay time on plus cams 9 – 16 |
|---|---|
| Meaning: | A lead or delay time can be assigned to each **plus cam 9 –16** in this setting data to compensate for delay times.<br><br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br><br>Positive value: Lead time<br>Negative value: Delay time<br><br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 9, 10, ... , 16<br><br>This setting data is added to MD: SW_CAM_PLUS_LEAD_TIME[n+8].<br><br>**SW 6.3 and higher:** The time settings act as the pulse duration on position-time cams. |
| Related to .... | MD: SW_CAM_PLUS_LEAD_TIME[n] (lead or delay time on plus cams 1 – 32) |

| **41524**<br>SD number | **SW_CAM_MINUS_TIME_TAB_3[n]**<br>Lead or delay time on minus cams 17 – 24 | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** | |
| Changes effective immediately | Protection level: 7/7 | Unit: s | |
| Data type: DOUBLE | Applies from SW: 4.1 | | |
| Meaning: | A lead or delay time can be assigned to each **minus cam 17 –24** in this setting data to compensate for delay times.<br><br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br><br>Positive value: Lead time<br>Negative value: Delay time<br><br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 17, 18, ... , 24<br><br>This setting data is added to MD: SW_CAM_MINUS_LEAD_TIME[n+16].<br><br>**SW 6.3 and higher:** The time settings act as a displacement in the trigger position on position-time cams. | | |
| Related to .... | MD: SW_CAM_MINUS_LEAD_TIME[n] (lead or delay time on minus cams 1 – 32) | | |

| 41525<br>SD number | SW_CAM_PLUS_TIME_TAB_3[n]<br>Lead or delay time on plus cams 17 – 24 | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** | |
| Changes effective immediately | Protection level: 7/7 | Unit: s | |
| Data type: DOUBLE | Applies from SW: 4.1 | | |
| Meaning: | A lead or delay time can be assigned to each **plus cam 17 –24** in this setting data to compensate for delay times.<br><br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br><br>Positive value: Lead time<br>Negative value: Delay time<br><br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 17, 18, ... , 24<br><br>This setting data is added to MD: SW_CAM_PLUS_LEAD_TIME[n+16].<br><br>**SW 6.3 and higher:** The time settings act as the pulse duration on position-time cams. | | |
| Related to .... | MD: SW_CAM_PLUS_LEAD_TIME[n] (lead or delay time on plus cams 1 – 32) | | |

| 41526<br>SD number | SW_CAM_MINUS_TIME_TAB_4[n]<br>Lead or delay time on minus cams 25 – 32 | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** | |
| Changes effective immediately | Protection level: 7/7 | Unit: s | |
| Data type: DOUBLE | Applies from SW: 4.1 | | |
| Meaning: | A lead or delay time can be assigned to each **minus cam 25 –32** in this setting data to compensate for delay times.<br><br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br><br>Positive value: Lead time<br>Negative value: Delay time<br><br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 7 corresponds to cam pair 25, 26, ... , 32<br><br>This setting data is added to MD: SW_CAM_MINUS_LEAD_TIME[n+24].<br><br>**SW 6.3 and higher:** The time settings act as a displacement in the trigger position on position-time cams. | | |
| Related to .... | MD: SW_CAM_MINUS_LEAD_TIME[n] (lead or delay time on minus cams 1 – 32) | | |

| | | | | | |
|---|---|---|---|---|---|
| **41527**<br>SD number | **SW_CAM_PLUS_TIME_TAB_4[n]**<br>Lead or delay time on plus cams 25 – 32 | | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** | |
| Changes effective immediately | | | Protection level: 7/7 | | Unit: s |
| Data type: DOUBLE | | | | Applies from SW: 4.1 | |
| Meaning: | A lead or delay time can be assigned to each **plus cam 25 –32** in this setting data to compensate for delay times.<br><br>The switching edge of the associated cam signal is advanced or delayed by the time value entered.<br><br>Positive value: Lead time<br>Negative value: Delay time<br><br>Index [n] of the setting data addresses the cam pair:<br>n = 0, 1, ... , 32 corresponds to cam pair 25, 26, ... , 32<br><br>This setting data is added to MD: SW_CAM_PLUS_LEAD_TIME[n+24].<br><br>**SW 6.3 and higher:** The time settings act as the pulse duration on position-time cams. | | | | |
| Related to .... | MD: SW_CAM_PLUS_LEAD_TIME[n] (lead or delay time on plus cams 1 – 32) | | | | |

■

# Notes

# Signal Descriptions 5

## 5.1 Signal overview

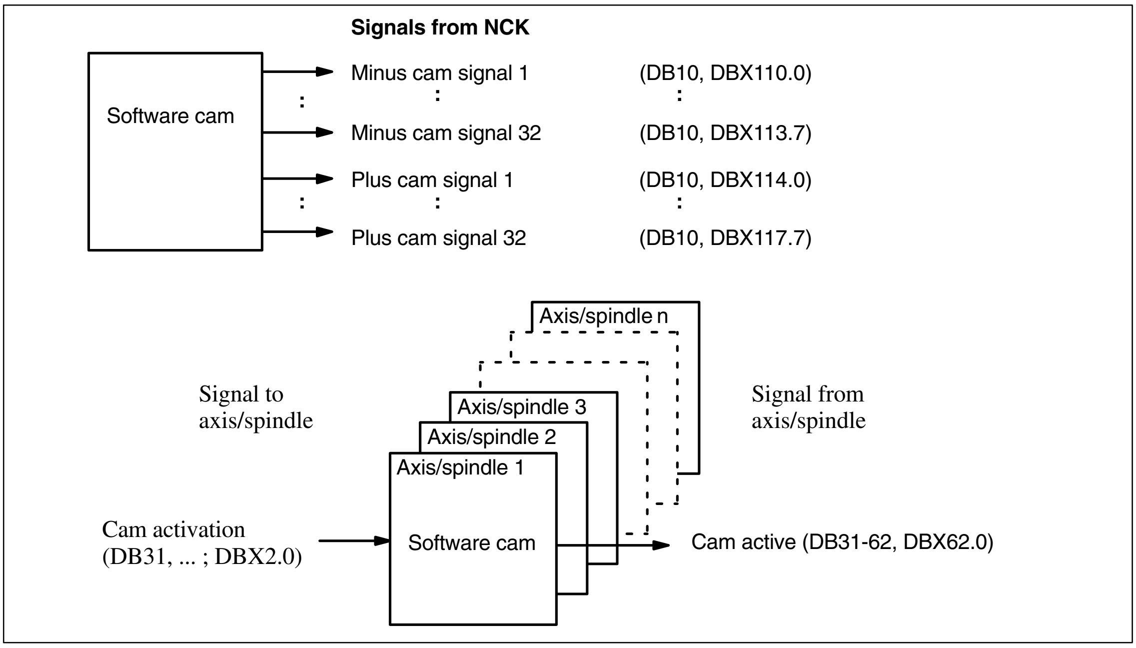



Fig. 5-1 PLC interface signals for “Software cams, position switching signals”

# 5.2 General signals

## 5.2.1 Signals from NCK

<table>
<tr><td>DB10<br>DBX110.0–113.7<br>Data Block</td><td colspan="2">Minus cam signals 1–32<br><br>Signal(s) from NCK (NCK → PLC)</td></tr>
<tr><td>Edge evaluation: no</td><td>Signal(s) updated: Cyclic</td><td>Signal(s) valid from SW: 2.1</td></tr>
<tr><td>Signal state 1 or signal transition 0 ––> 1</td><td colspan="2">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.<br><br>Linear axis:<br>– The minus cam signal switches from 0 to 1 if the axis overtravels the minus cam in the negative axis direction.<br><br>Modulo rotary axis:<br>– The minus cam signal changes level in response to every positive edge of the plus cam signal.</td></tr>
<tr><td>Signal state 0 or signal transition 1 ––> 0</td><td colspan="2">Linear axis:<br>– The minus cam signal switches from 1 to 0 when the axis traverses the minus cam in the positive axis direction.<br><br>Modulo rotary axis:<br>– The minus cam signal changes level in response to every positive edge of the plus cam signal.</td></tr>
</table>

<table>
<tr><td>DB10<br>DBX114.0–117.7<br>Data Block</td><td colspan="2">Plus cam signals 1–32<br><br>Signal(s) from NCK (NCK → PLC)</td></tr>
<tr><td>Edge evaluation: no</td><td>Signal(s) updated: Cyclic</td><td>Signal(s) valid from SW: 2.1</td></tr>
<tr><td>Signal state 1 or signal transition 0 ––> 1</td><td colspan="2">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.<br><br>Linear axis:<br>– The plus cam signal switches from 0 to 1 when the axis traverses the plus cam in the positive direction.<br><br>Modulo rotary axis:<br>– The plus cam signal switches from 0 to 1 when the minus cam is overtraveled in the positive axis direction.<br><br>The described response of the plus cam applies under the condition:<br><br>Plus cam – minus cam < 180 degrees<br><br>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.</td></tr>
<tr><td>Signal state 0 or signal transition 1 ––> 0</td><td colspan="2">Linear axis:<br>– The plus cam signal switches from 1 to 0 if the axis overtravels the plus cam in the negative direction.<br><br>Modulo rotary axis:<br>– The plus cam signal switches from 1 back to 0 if the plus cam is overtraveled in the positive axis direction.<br><br>The described response of the plus cam applies under the condition:<br><br>Plus cam – minus cam < 180 degrees<br><br>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.</td></tr>
</table>

# 5.3 Axis/spindle-specific signals

## 5.3.1 Signals to axis/spindle

| **DB31-62**<br>**DBX2.0**<br>Data Block | **Cam activation**<br><br>Signal(s) to axis/spindle (PLC → NCK) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ——> 1 | Output of the minus and plus cam signals of an axis to the general PLC interface is activated.<br>The activation takes effect immediately after processing of IS “Cam activation” in the NCK. | |
| Signal state 0 or signal transition 1 ——> 0 | The minus and plus cam signals of an axis are **not** output to the general PLC interface. | |
| Related to .... | IS “Minus cam signal 1–32” (DB10, DBX110.0–113.7)<br>IS “Plus cam signal 1–32” (DB10, DBX114.0–117.7) | |

## 5.3.2 Signals from axis/spindle

| **DB31-62**<br>**DBX62.0**<br>Data Block | **Cams active**<br><br>Signal(s) from axis/spindle (NCK → PLC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ——> 1 | All cams of the axis selected via IS “Cam activation” (DB31–48, DBX2.0) have been activated successfully. | |
| Signal state 0 or signal transition 1 ——> 0 | The cams of the axis are **not** activated. | |
| Related to .... | IS “Cam activation” (DB31–62, DBX2.0)<br>IS “Minus cam signal 1–32” (DB10, DBX110.0–113.7)<br>IS “Plus cam signal 1–32” (DB10, DBX114.0–117.7) | |

■

# Notes

# 6 Example

– None –

# 7 Data Fields, Lists

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **General** | | | |
| 10 | 110.0...110.7 | Minus cam signal 1...8 | |
| 10 | 111.0...111.7 | Minus cam signal 9...16 | |
| 10 | 112.0...112.7 | Minus cam signal 17...24 | |
| 10 | 113.0...113.7 | Minus cam signal 25...32 | |
| 10 | 114.0...114.7 | Plus cam signal 1...8 | |
| 10 | 115.0...115.7 | Plus cam signal 9...16 | |
| 10 | 116.0...116.7 | Plus cam signal 17...24 | |
| 10 | 117.0...117.7 | Plus cam signal 25...32 | |
| **Axis-specific** | | | |
| 31, ... ; | 2.0 | Cam activation | |
| 31, ... ; | 62.0 | Cams active | |

## 7.2 Machine Data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ... )** | | | |
| 10260 | CONVERT_SCALING_SYSTEM | Basic system switchover active | G2 |
| 10270 | POS_TAB_SCALING_SYSTEM | System of measurement of position tables | T1 |
| 10450 | SW_CAM_ASSIGN_TAB[n] | Assignment of software cams to machine axes | |

| General ($MN_ ... ) | | | |
|---|---|---|---|
| 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 | |

## 7.3 Setting data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($SN_ ...)** | | | |
| 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 | |

## 7.4 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, “Diagnostics Guide”
and the online help of MMC 101/102/103 systems.

■

# Notes

# SINUMERIK 840D/840Di/810D Description of Functions Extended Functions (FB2)

# Punching and Nibbling (N4)

■

# 1 Brief Description

The functions specific to punching and nibbling operations comprise the following:

- Stroke control
- Automatic path segmentation
- Rotatable punch and die
- Clamp protection

including their activation and deactivation via language commands.

■

# Notes

# Detailed Description 2

## 2.1 Stroke control

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.

### 2.1.1 High-speed signals

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 operate the punching unit.
The following diagram shows the signal sequence.

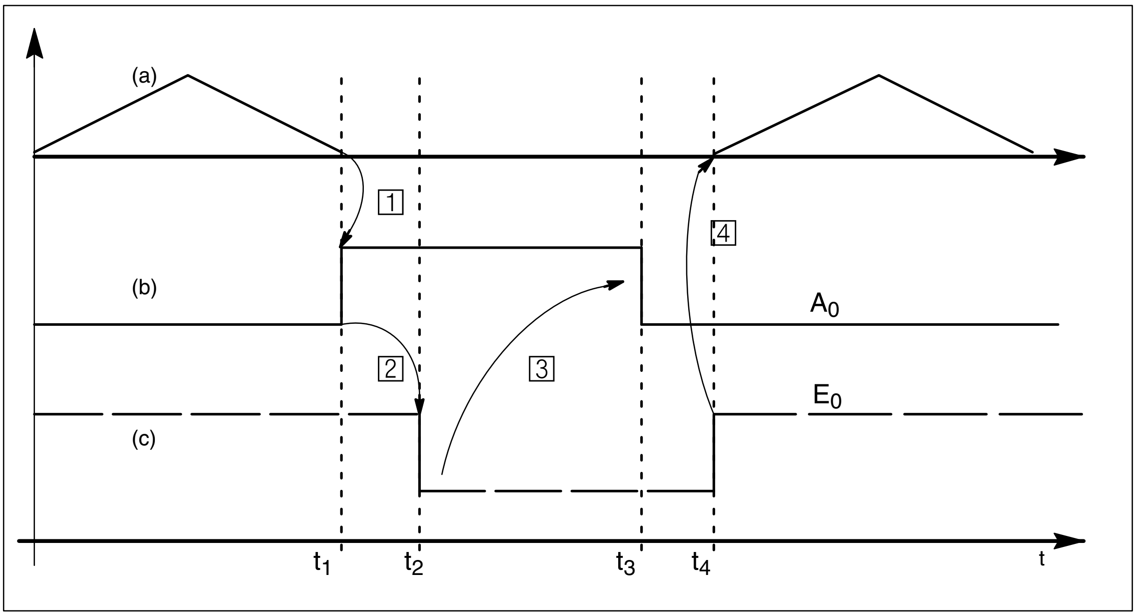



Fig. 2-1 Signal chart

---

**Note**

The diagram illustrates the following:

(a) Axis motion of the machine as a v(t) function
(b) “Stroke activation” signal
(c) “Stroke active” signal

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 $E_0$.
$A_0$ is set by the NCK and is identical to stroke initiation.
$E_0$ defines the status of the punching unit and is 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 activation (see following section “Criteria for stroke initiation”), high-speed output $A_0$ is set for punch initiation $\boxed{1}$.

**$t_2$**

The punching unit signals a punch movement via high-speed input $E_0$ at instant $t_2$. This is triggered by signal A0 $\boxed{2}$.

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 activation” signal $\boxed{3}$. 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 $\boxed{4}$. The reaction of the NC to a signal edge change $\boxed{4}$ is described in the section headed “Axis start after punching” below.

---

**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.

---

## 2.1.2 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.

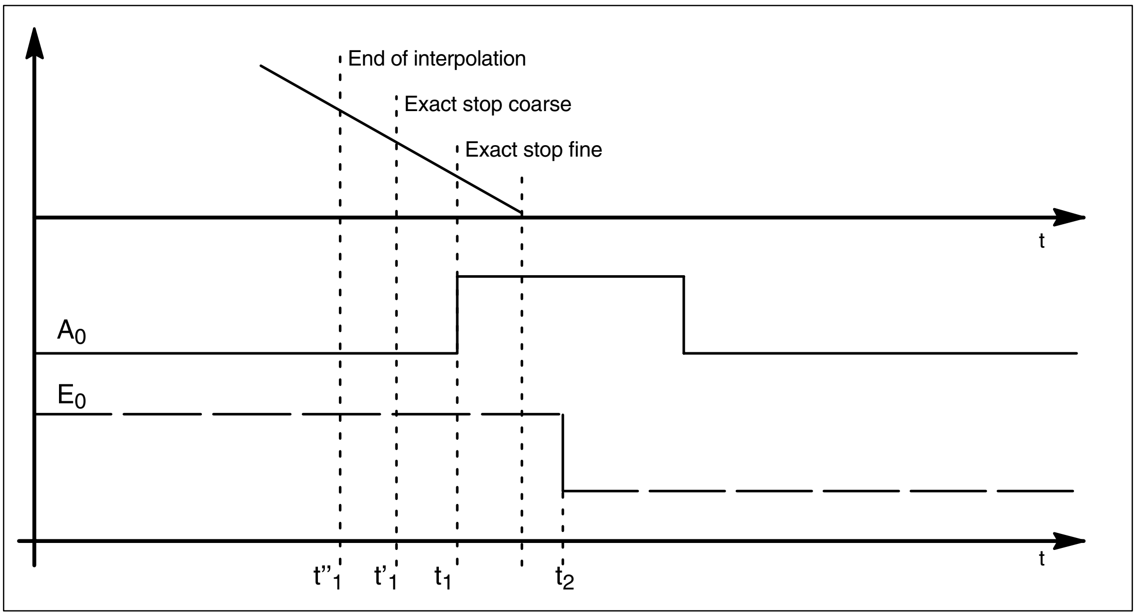



Fig. 2-2 Signal chart: Criteria for stroke initiation

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 (default setting of G group 12 G601). The punch initiation times t"$_1$ and t'$_1$ are programmed by means of G602 and G603 (see table below).

| If | Then | Description |
|---|---|---|
| G603 is programmed, | Stop the 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"$_1$). |
| G602 is programmed, | 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 is programmed, | 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 well.<br>However, this variant also results in a maximum dead time (see $t_1$). |
| **Note:**<br>The initial setting of the G group with G601, G602 and G603 (G group 12) is defined via MD: GCODE_RESET_VALUES[11] (G601 is the default setting) | | |

**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.

In combination with machine data 26018: NIBBLE_PRE_START_TIME, 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".

Apart from MD 26018: NIBBLE_PRE_START_TIME, SD 42402 NIB_PUNCH_PRE_START_TIME is also available. This can be altered from the parts program and thus adapted to the punching process depending on the processing status of the parts program.

The following delay times apply depending on the value programmed for the setting data:

MD 26018 $=$ 0 $\rightarrow$ SD 42402 is active.
MD 26018 $\neq$ 0 $\rightarrow$ MD 26018 is active.

If the “Punching with dwell time, PDELAYON” is active, then the dwell time programmed in connection with this function is active. Both MD 26018 and SD42402 are inoperative.

### 2.1.3 Axis start after punching

The start of an axis motion after stroke initiation is controlled via input signal “Stroke ON”.

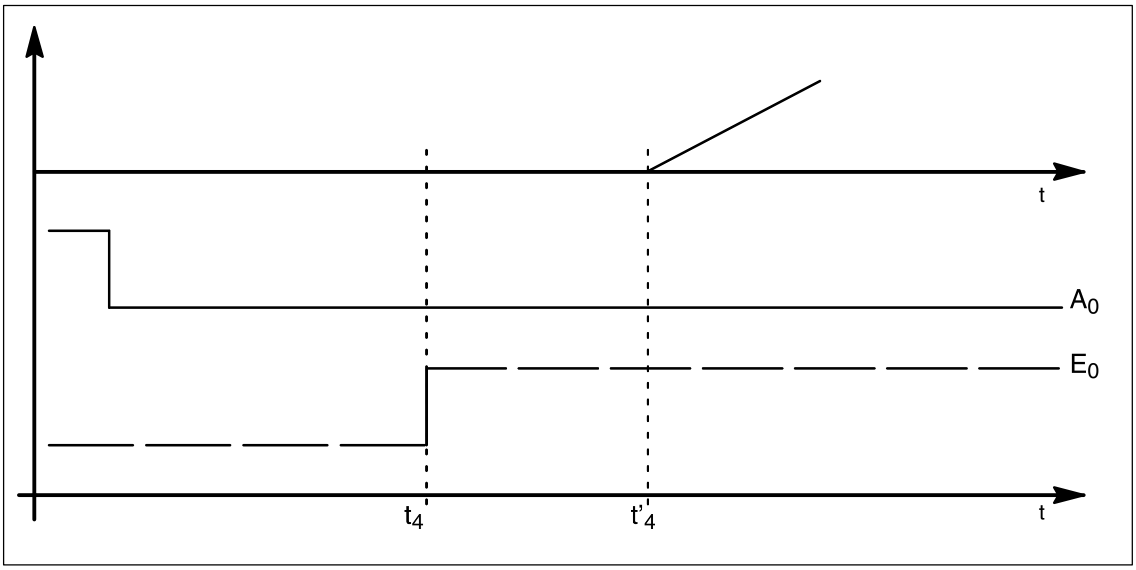



Fig. 2-3 Signal chart: Axis start after punching

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 by means of PON/SON, the maximum delay time is calculated as $|t'_4 - t_4|$ = 3 x interpolation cycle.

**PONS/SONS**

If the punch is controlled by means of PONS/SONS, then the delay time is determined by $|t'_4 - t_4| \leq$ 3 x position control cycle.
(Precondition: Stroke time $(t_4 - t_2)$ > 4 interpolation cycles.)

### 2.1.4 PLC signals specific to punching and nibbling

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.

The “No stroke enable” signal prevents the NC from initiating any punching operation. The NC waits until the enable signal is available before continuing the parts program. The “Stroke suppression” signal allows the parts program to be processed without initiating a punching operation (dry run). With active path segmentation, the axes traverse in “Stop and go” mode. The “Delayed stroke” signal activates a delayed stroke output such as that programmable with PDELAYON. The “Manual stroke initiation” signal allows the operator to initiate a punching process (controlled via PLC), even when the parts program is not being processed. This signal is acknowledged by the “Acknowledge manual stroke initiation” signal.

---

**Note**

The signals from/to channel are described in Chapter 5 and are listed in Chapter 7.

---

### 2.1.5 Punching and nibbling-specific reactions to standard PLC signals

**“Feed stop” interface signal**

In the case of a “Feed stop” signal, the NC reacts as follows with respect to the stroke control:
If the signal is detected before instant $t_1$, then 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. If the signal is detected at instant $t_1$, then the current stroke is completed and the NC then rests 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.

### 2.1.6 Signal monitoring

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 such cases, an alarm may be generated depending on machine data 26020: NIBBLE_SIGNAL_CHECK (alarm 22054 “distorted punch signal”).

**Reset response**

In the case of an NC RESET, the “Stroke initiation” signal is canceled immediately even if the acknowledgement via the high-speed input has not arrived. A currently activated stroke cannot be suppressed.

## 2.2 Activation and deactivation

### 2.2.1 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:** /PA/, Programming Guide

**Groups**

The language commands are arranged in groups as follows:

**Group 35**
The actual punching and nibbling functions are activated and deactivated by means of the following language commands:

PON = Punching ON
SON = Nibbling ON
PONS = Punching ON, activation in position controller
SONS = Nibbling ON, activation in 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. These language commands are as follows:

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 is equivalent to a RESET state.

Programming example:

```
:
:
N20   G90   X100        SON    Activate punching
N25   X50   SPOF               Deactivate punching,
:                              Positioning without stroke initiation
:
```

**SON**

**Nibbling ON**
SON activates the nibbling function and deselects the other functions in G group 35 (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:

```
:
:
N70   X50   SPOF               Positioning without punch initiation
N80   X100  SON                Activate nibbling, initiate a stroke
:                              before motion (X=50) and at end of
:                              programmed motion (X=100)
```

**SONS**

**Nibbling ON (in position control cycle)**
SONS acts 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.
Like SON, PON also has a modal action.
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:

```
:
:
N100  Y30   SPOF               Positioning without punch initiation
N110  Y100  PON                Activate punching, initiate punch
:                              at end of positioning operation (Y=100)
:
```

**PONS**

**Punching ON (in position controller)**
PONS acts 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 can be defined in seconds by setting data 42400: 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 machine data 26004: NIBBLE_PUNCH_OUTMASK and MD 26006: NIBBLE_PUNCH_INMASK):

```
:
N170  PDELAYON    X100    SPOF   Positioning without stroke initiation,
:                                activate delayed punch
:                                initiation
N180  X800  PON                  Activate punching. When end
:                                position is reached, a punch stroke
:                                is output with a delay.
N190  PDELAYOF    X700           Deactivate delayed
:                                punching, activate normal
:                                punch initiation. End of
:                                programmed motion.
```

**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 machine data 26004: NIBBLE_PUNCH_OUTMASK and MD 26006: NIBBLE_PUNCH_INMASK). 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 machine data 26004: NIBBLE_PUNCH_OUTMASK and MD 26006: NIBBLE_PUNCH_INMASK).

Programming example:

| | | | | |
|---|---|---|---|---|
| : | | | | |
| : | | | | |
| N170 | SPIF1 | X100 | PON | At the end of the block a stroke is |
| : | | | | initiated at the first high-speed |
| : | | | | output. The “Stroke active” signal is |
| : | | | | monitored at the first output. |
| N180 | X800 SPIF2 | | | The second stroke is initiated at |
| : | | | | the second high-speed output. |
| : | | | | The “Stroke active” signal is |
| : | | | | monitored at the second input. |
| N190 | SPIF1 | X700 | | The first interface is used to |
| : | | | | control all further strokes. |

### 2.2.2 Expansions to punching and nibbling functions

**Alternate interface in SW 3.2 and higher**

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 machine data

MD 26004: NIBBLE_PUNCH_OUTMASK and
MD 26006: NIBBLE_PUNCH_INMASK.

The alternate interfaces can be selected via commands SPIF1 and SPIF2. Full punching/nibbling functionality is available on both interfaces.

Example:

Hardware assignment for stroke control
Define the fast byte in each case on the CPU as a fast punching interface:
$MC_PUNCHNIB_ASSIGN_FASTIN = 'H00030001' →byte 1
$MC_PUNCHNIB_ASSIGN_FASTOUT = 'H00000001'
Note: From SW 3.2 First and second bit inverted
Inverted directly by the software in SW version 3.1 and earlier.

Mask for fast output and input bits

| | |
|---|---|
| | First interface output bit |
| MD 26004: NIBBLE_PUNCH_OUTMASK[0] = 1 | → Bit 1 SPIF1 |
| | Second interface output bit |
| MD 26004: NIBBLE_PUNCH_OUTMASK[1] = 2 | → Bit 2 SPIF2 |
| | First interface output bit |
| MD 26006: NIBBLE_PUNCH_INMASK[0] = 1 | → Bit 1 SPIF1 |
| | Second interface output bit |
| MD 26006: NIBBLE_PUNCH_INMASK[1] = 2 | → Bit 2 SPIF2 |

**Automatically activated pre-initiation time in SW 3.1 and higher**

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 machine data MD 26018: NIBBLE_PRE_START_TIME.

A value of 0.010 s is selected to initiate a stroke, for example, 2 cycles after the end of interpolation with an interpolation cycle of 5 ms.

The pre-initiation time can be set in SD 42402: NIBPUNCH_PRE_START_TIME. This setting becomes operative only if MD 26018: NIBBLE_PRE_START_TIME is set to 0. This then gives higher priority to the pre-initiation time in MD 26018: NIBBLE_PRE_START_TIME.

**Monitoring of the input signal in SW 3.1 and higher**

If the “stroke active” signal is fluctuating between strokes due to plunger overshoots, for example, the message “undefined punching signal” is output and interpolation halted. This message is generated as a function of machine data MD 26020: NIBBLE_SIGNAL_CHECK.

MD 26020: NIBBLE_SIGNAL_CHECK = 0 No alarm.

**Minimum time between two strokes in SW 5.2 and higher**

Setting data SD 42404: MINTIME_BETWEEN_STROKES can be parameterized to set a minimum time interval between two consecutive strokes. If, for example, an interval of at least 1.3 seconds must elapse between two stroke initiations, irrespective of physical distance, then

SD 42404: MINTIME_BETWEEN_STROKES must be set to 1.3.

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: MINTIME_BETWEEN_STROKES has elapsed.

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 can be taken to prevent this:

Example:
N...
N100 STOPRE
N110 $SC_ MINTIME_BETWEEN_STROKES = 1.3

The function is not active when $SC_MINTIME_BETWEEN_STROKES = 0.

**Path-dependent acceleration in SW 5.2 and higher**

An acceleration characteristic can be defined by means of language command PUNCHACC($S_{min}$, $A_{min}$, $S_{max}$, $A_{max}$). This command can be used to define different acceleration rates depending on the distance between holes.

The characteristic shown in Fig. 2-4 defines the following acceleration rate:

- Distance between holes less than 2 mm:
  The axis accelerates at a rate corresponding to 50% of maximum acceleration.
- Distance between holes from 2 mm to 10 mm:
  The acceleration rate is increased to 100% as a proportion of distance.
- Distance between holes more than 10 mm:
  The axis accelerates at a rate corresponding to 100% of maximum acceleration.

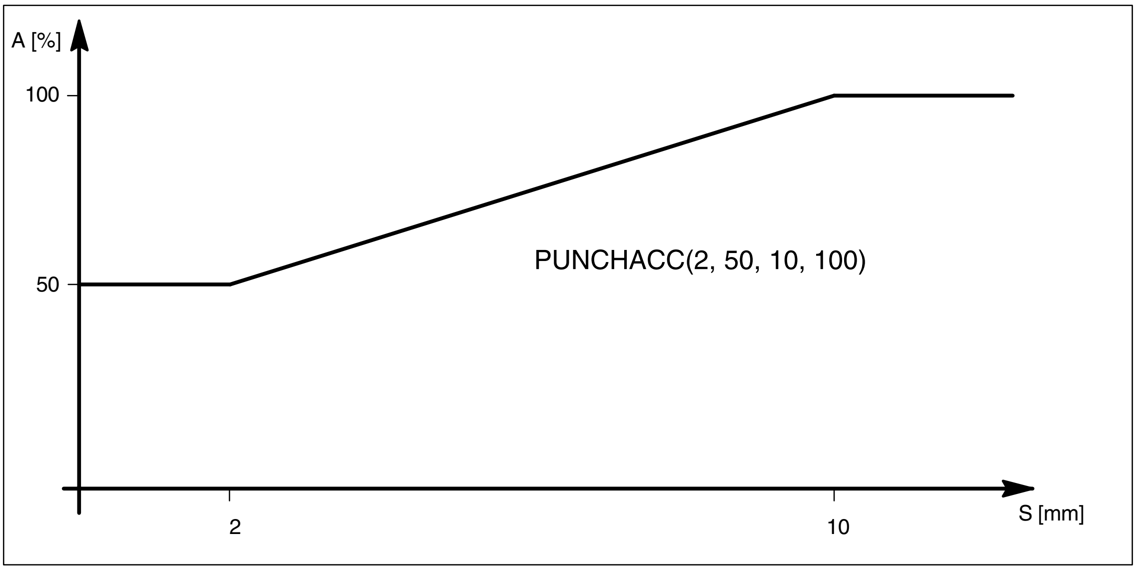



Fig. 2-4 Proportional increase in acceleration from 50% with a distance between holes of 2 mm to 10 mm

The characteristic shown in Fig. 2-5 defines the following acceleration rate:

- Distance between holes less than 3 mm:
  The axis accelerates at a rate corresponding to 75% of maximum acceleration.
- Distance between holes from 3 mm to 8 mm:
  The acceleration rate is reduced to 25% as a proportion of distance.
- Distance between holes more than 8 mm:
  The axis accelerates at a rate corresponding to 25% of maximum acceleration.

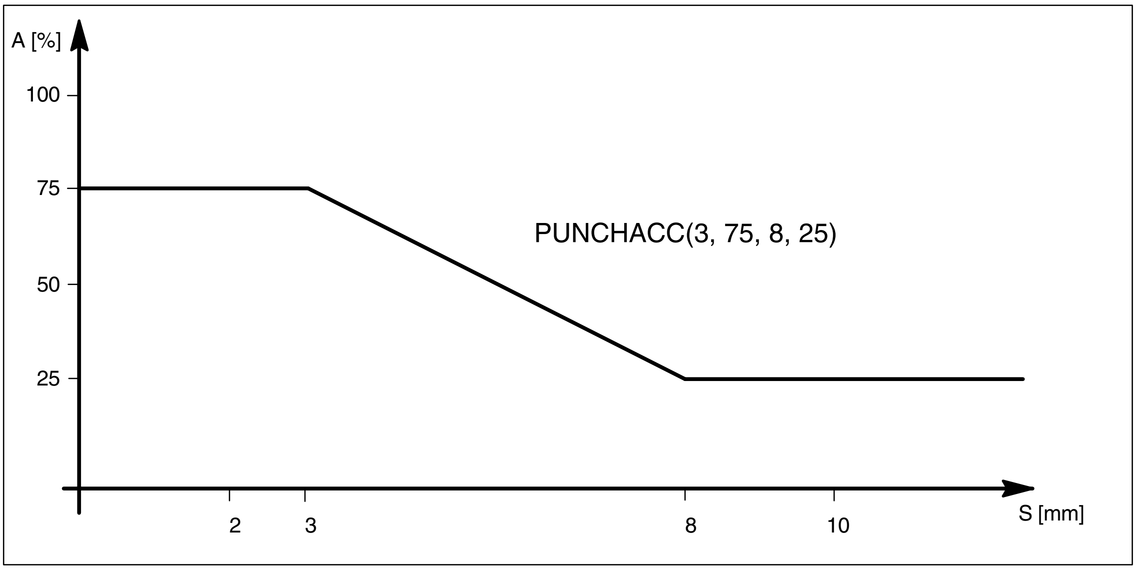



Fig. 2-5 Proportional reduction in acceleration from 75% with a distance between holes of 3 mm to 8 mm

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 existing acceleration programming with ACC remains operative.

---

**Note**

For further details, please refer to:

| | |
|---|---|
| Chapter 4 | 4.1 “Channel-specific machine data” and<br>4.2 “Channel-specific setting data” |

---

### 2.2.3 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 following assignments corresponding to those used in earlier systems apply:

M20, M23 ≙ SPOF
M22 ≙ SON
M25 ≙ PON
M26 ≙ PDELAYON

---

**Note**

The M functions can be configured via machine data.
As regards the assignments between the M functions and language commands, it must be noted that the M functions are divided into auxiliary function groups.

---

**Examples**

| | |
|---|---|
| DEFINE M20 AS SPOF or<br>DEFINE M20 AS SPOF M=20 | Punching/nibbling OFF<br>Punching with auxiliary function output |
| DEFINE M20 AS SPOF PDELAYOF | Punching/nibbling OFF and delayed punching OFF |
| DEFINE M22 AS SON or<br>DEFINE M22 AS SON M=22 | Nibbling ON<br>Nibbling ON with auxiliary function output |
| DEFINE M25 AS PON or<br>DEFINE M25 AS PON M=25 | Punching ON<br>Punching ON with auxiliary function output |
| DEFINE M26 AS PDELAYON or<br>DEFINE M26 AS PDELAYON M=26 | Delayed punching<br>Punching and auxiliary function output |

Programming example:

```
:
:
N100  X100  M20               Positioning without punch initiation
N110  X120  M22               Activate nibbling, stroke initiation before
:                             and after motion
N120  X150  Y150        M25   Activate punching,
:                             stroke initiation at end of motion.
:
```

## 2.3 Automatic path segmentation

One of the following two methods can be applied to automatically segment a programmed traversing path:

- Path segmentation with maximum path feed value programmed via language command SPP and
- Path segmentation with a number of segments programmed via language command SPN.

Both functions generate sub-blocks independently. In earlier systems, language command SPP <number> corresponds to E <number> and SPN <number> 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 punching stroke is made at the end of this sub-block.

SPP and SPN can be activated only if geo axes are configured.

---

**SPP**

- The automatic path segmentation function SPP divides the programmed traversing path into sections of equal size as a function of the programmed feed path.
- Path segmentation is active only when SON or PON is active. (Exception: MD 26014: PUNCH_PATH_SPLITTING = 1).
- SPP is modally active, i.e. the programmed feed value 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 feed value 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. SPP 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.
- SPN is active non-modally and is activated if SON or PON has already been activated.
(Exception: MD 26014: PUNCH_PATH_SPLITTING = 1).
- Any previously programmed SPP value is suppressed for the block containing SPN, but is re-activated again in the following blocks.

**Supplementary conditions**

- Path segmentation is active for linear and circular interpolation.
The interpolation mode remains valid, i.e. circles are traversed in the case of circular interpolation.
- 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: MD 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 same rules apply to programmed SPP and SPN commands as defined for SON and PON, i.e. a stroke is initiated only if an axis motion has been programmed.

## 2.3.1 Operating characteristics with path axes

All axes defined and programmed via machine data PUNCHNIB_AXIS_MASK (26010) 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 (see below).

**Example of SPP**

```
N1  G01   X0    Y0       SPOF  Position without punch initiation
N2  X75   SPP=25         SON   Nibble at feed value 25 mm;
:                              Initiate punch before first
:                              motion and after each path segment.
N3  Y10                        Position with reduced SPP value
:                              because traversing distance <
:                              SPP value and initiate punch after
:                              motion.
N4  X0                         Reposition with initiation of punch
:                              after every path segment.
```

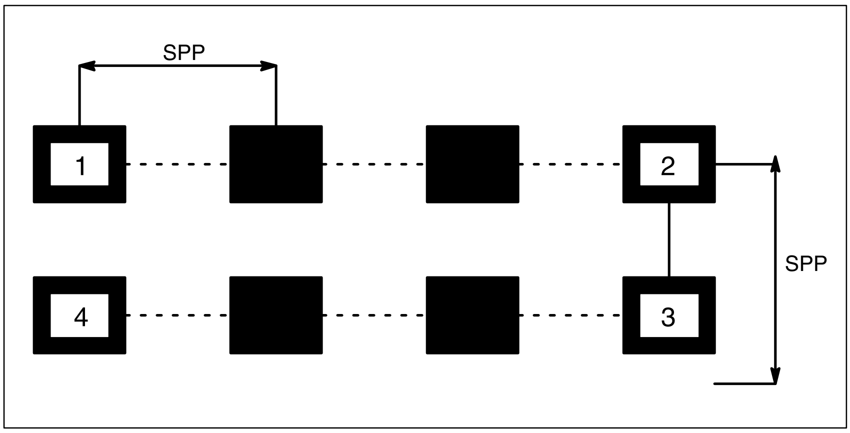



Fig. 2-6

If the programmed path segmentation is not an integral multiple of the total path, then the feed path is reduced (see following diagram).

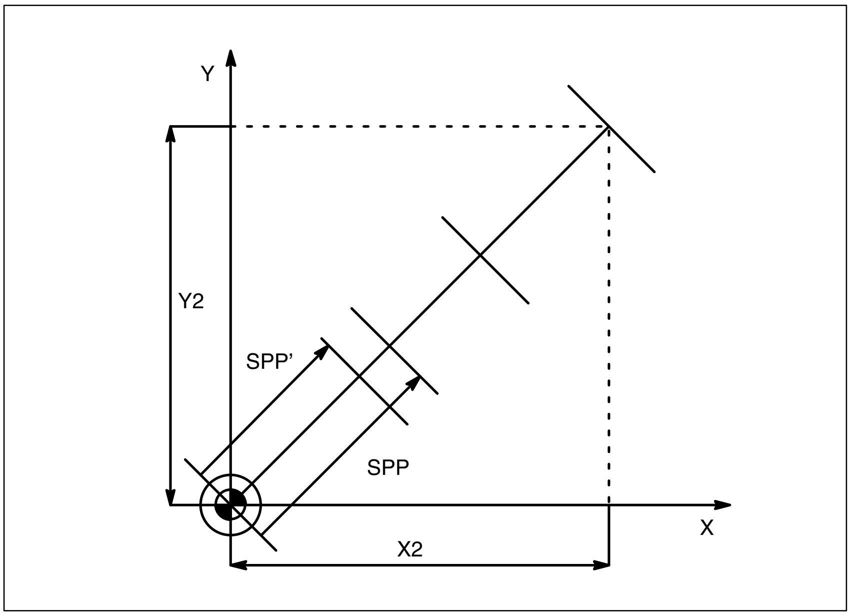



Fig. 2-7 Path segmentation

| | |
|---|---|
| X2/Y2 | Programmed traversing path |
| SPP | Programmed SPP value |
| SPP' | Automatically rounded offset path |

**Example of SPN command**

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. In the case of nibbling operations, this is executed at the beginning of the first segment. 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.

```
N1   G01   X0       Y0     SPOF   Position without initiation of punch,
N2   X75   SPN=3    SON           Activate nibbling, the whole path
:                                 is divided into three segments.
:                                 Since nibbling is active, a stroke
:                                 is initiated before the first motion
:                                 and at the end of each segment.
N3   Y10   SPOF                   Position without initiation of punch
N4   X0    SPN=2    PON           Activate punching, the whole path
:                                 is divided into two segments.
:                                 Since punching is active, the first
:                                 stroke is initiated at the end of the
:                                 first segment.
```

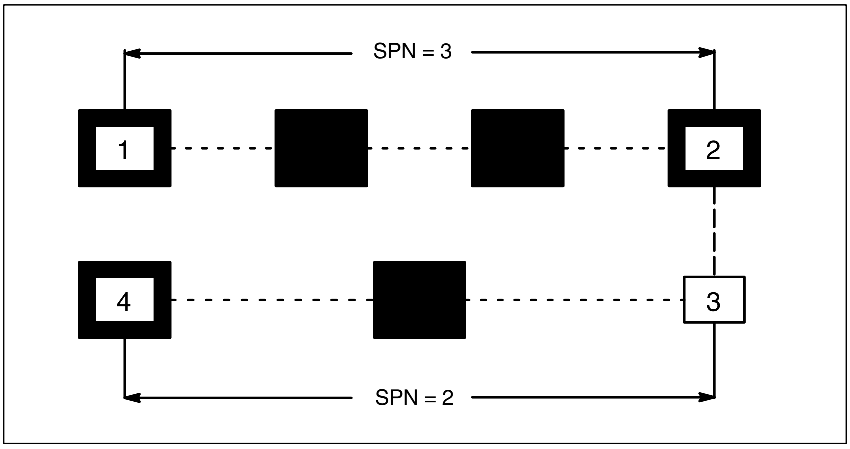



Fig. 2-8

## Example

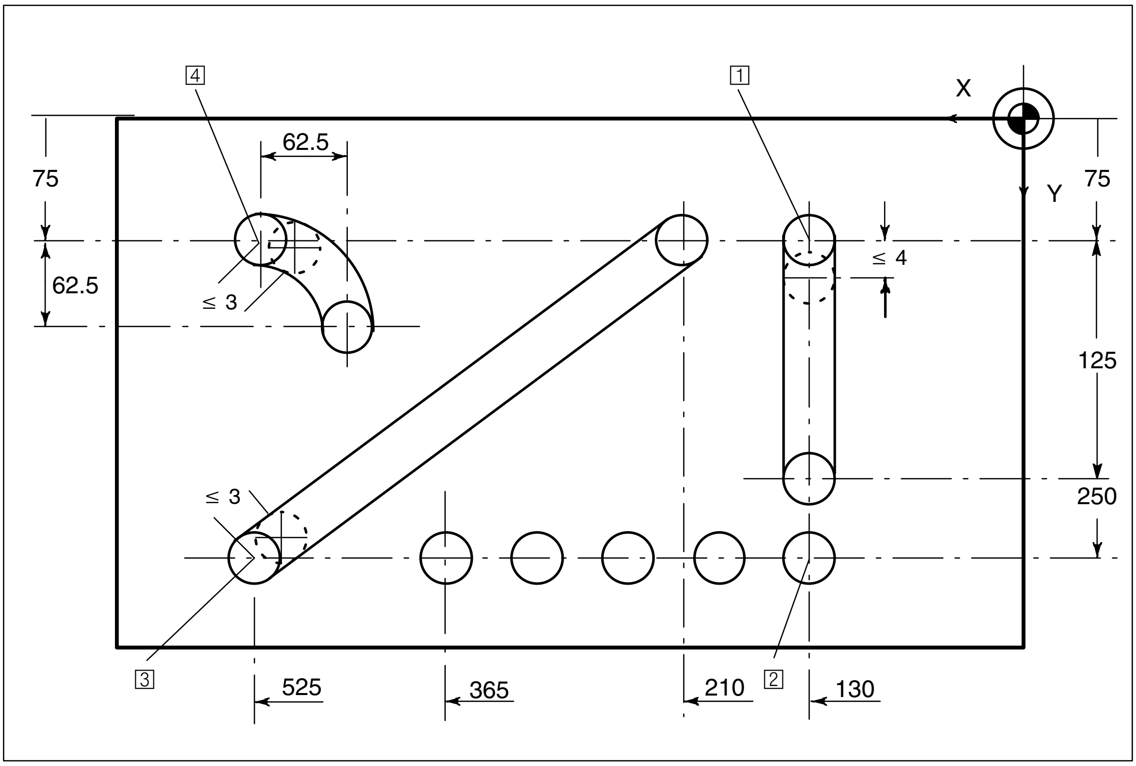



Fig. 2-9 Workpiece

**Extract from program**

| | |
|---|---|
| N100 G90 X130 Y75 F60 SPOF | Position at start point 1 of vertical nibble paths |
| N110 G91 Y125 SPP=4 SON | End point coordinates (incremental); feedrate value: 4 mm, activate nibbling |
| N120 G90 Y250 SPOF | Absolute dimensioning, Position at start point 2 of horizontal nibble path |
| N130 X365 SPP=4 SON | End point coordinates, 4 segments, activate nibbling |
| N140 X525 SPOF | Position at start point 3 of inclined nibble path |
| N150 X210 Y75 SPP=3 SON | End point coordinates feedrate value: 4 mm, Activate nibbling |
| N160 X525 SPOF | Position at start point 4 of nibble path on pitch circle path |
| N170 G02 G91 X–62.5 Y62.5 I0 J62.5 SON | Incremental circular interpolation with interpolation parameters, Activate nibbling |
| N180 G00 G90 Y300 SPOF | Positioning |

### 2.3.2 Response in connection with single axes

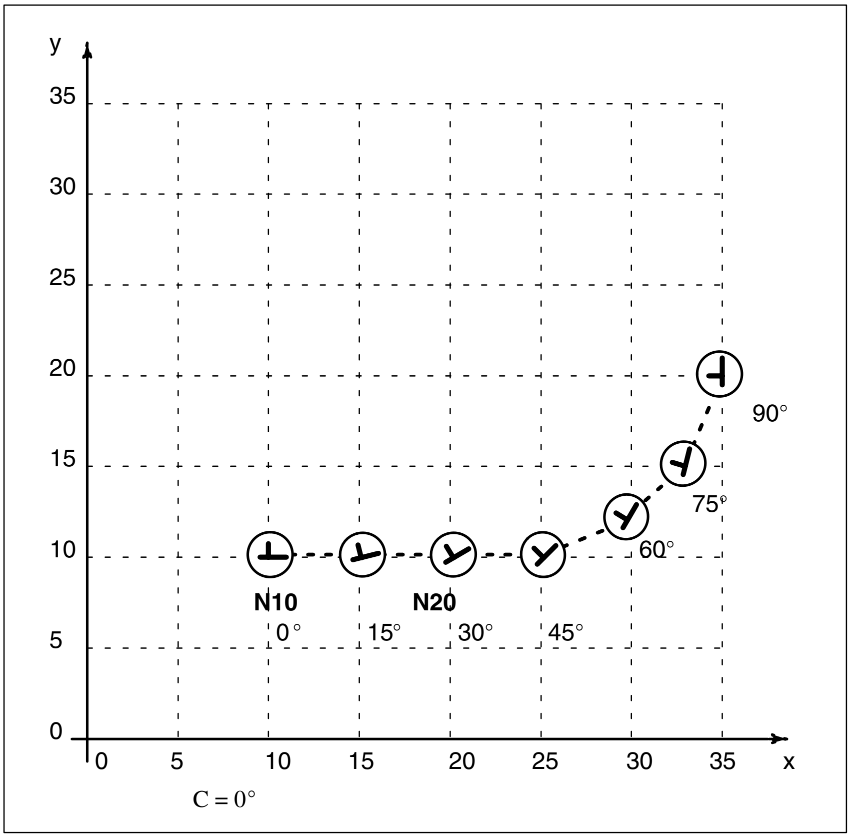
The path of single axes programmed in addition to path axes is distributed evenly among the generated intermediate blocks as standard. In the following example, the additional rotary axis C is defined as a synchronous axis in the system. If this axis is programmed as a “Punch-nibble axis” (via PUNCHNIB_AXIS_MASK = 1 for this axis), then the behavior of the synchronous axis can be varied as a function of machine data PUNCH_PARTITION_TYPE.

**Example**

```
N10   G1      PON  X10  Y10  C0
N20   SPP=5   X25  C45
N30   SPN=3   X35  Y20  I10  J10  C90
```

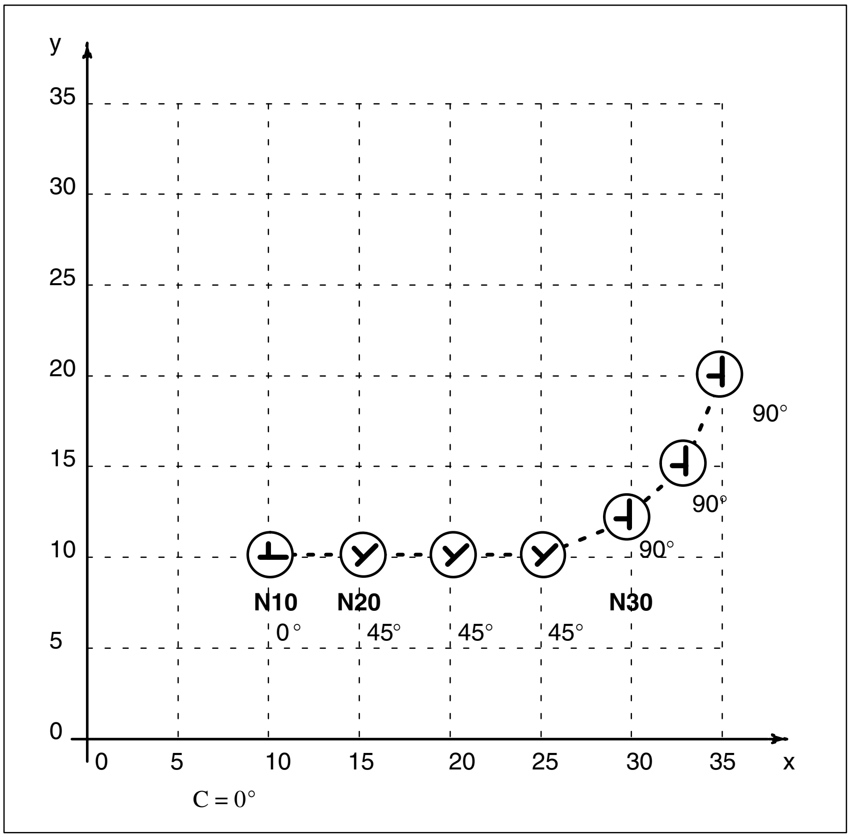
**PUNCH_PARTITION_TYPE=0** (default setting)

In the above example, 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).



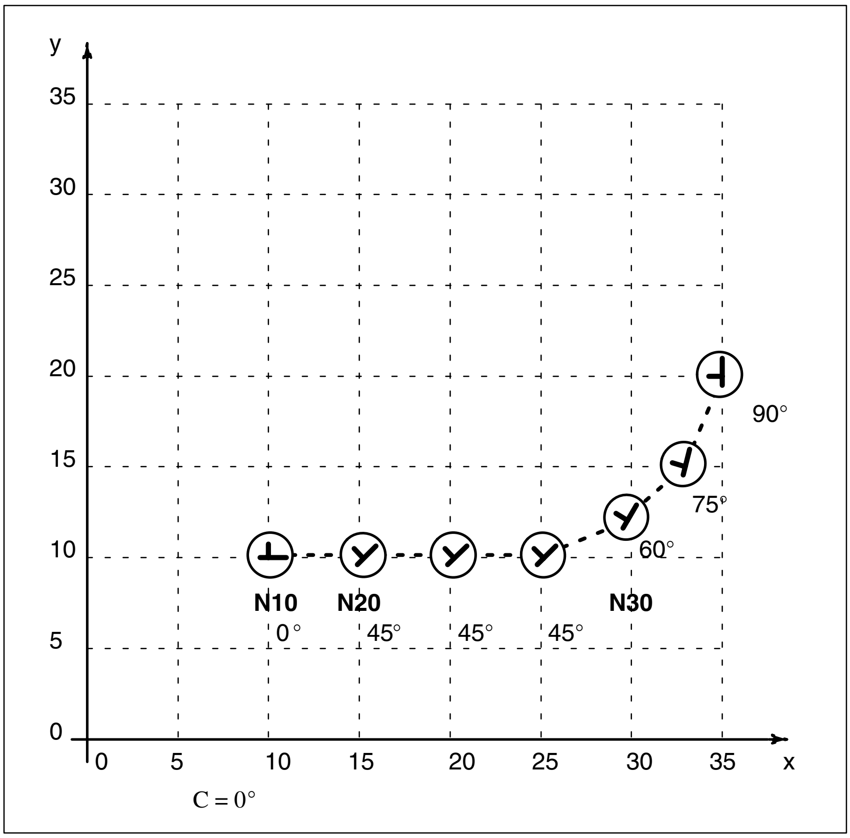
**PUNCH_PARTITION_TYPE=1**

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 above 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.




**PUNCH_PARTITION_TYPE=2**

MD=2 is set in cases where the axis must behave as described above (PUNCH_PARTITION_TYPE=1) in linear interpolation mode, but according to the default setting in circular interpolation mode (see 1st case). Given the above example, the axis then behaves as follows: In block N20, the C axis is rotated to C=45° in the first sub-block. The following circular 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.

---

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 PLC interface signal (stroke suppression). (Application: e.g. alignment of electron beam during welding).

Similar axis operating characteristics can be obtained by setting
MD 26014 PUNCH_PATH_SPLITTING to “1”.
In this case, the path is segmented irrespective of punching or nibbling functions.

## 2.4 Rotatable tool

The functions

- “Coupled motion” for synchronous rotation of punch and die and
- “Tangential control” for normal alignment of rotary axes for punching tools in relation to workpiece

can be used on nibbling/punching machines with rotatable punching tool and die to achieve a wide variety of applications for the punch.

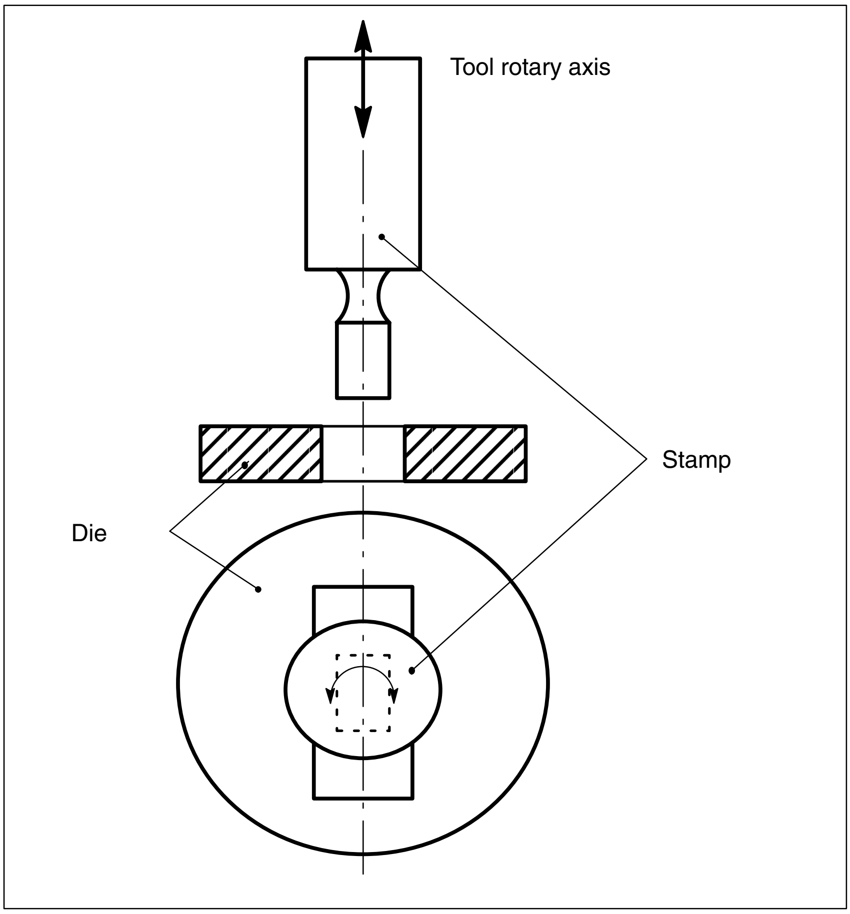



Fig. 2-10 Illustration of a rotatable tool axis

### 2.4.1 Coupled motion of punch and die

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 or TRAILOF.

**References:** /FB/, M3 “Coupled Motion”

**Example**

Example of a typical nibbling machine with rotatable punching tools where C is the punch axis and C1 the die axis

```
:
:
TRAILON (C, C1, 1);                     Switch on coupled motion grouping
G01   X100  Y100     C0        PON      Initiate stroke with C axis and
                                        C1 axis position C=0=C1
X150  C45                               Initiate stroke with C axis/C1 axis
:                                       position C=45=C1
:
M30
```

**Initial setting**

No coupled motion groupings are active after power-up. Once the two tool axes have approached the reference point, the coupled-axis grouping is not generally separated again. This can be achieved by activating the coupled-axis grouping once (see above example) and setting machine data 20110 RESET_MODE_MASK, bit 8=1. In this way, the coupled-axis grouping remains active after RESET/parts program start or end.

### 2.4.2 Tangential control

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 or TANGOF respectively.

**References:** /PA/, Programming Guide, Fundamentals

**Operating principle**

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 axes” can be used.
The tangential control function must be activated first followed by coupled axes.

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.

```
:
:
N2    TANG (C, X, Y, 1, “B”)
N5    G0    X10    Y5
N8    TRAILON (C, C1, 1)
N10   Y10   C225   PON      F60
N15   X20   Y20    C45
N20   X50   Y20    C90      SPOF
N25   X80   Y20    SPP=10   SON
N30   X60   Y40    SPOF
N32   TANGON (C, 180)
N35   X30   Y70    SPP=3    PON
N40   G91   C45    X–10     Y–10
N42   TANGON (C, 0)
N45   G90   Y30    SPP=3    SON
N50   SPOF  TANGOF
N55   M2
```

### Explanations

Table 2-1

| Set | Remarks |
|---|---|
| N2 | Definition of master and slave axes, C is slave axis for X and Y in the base coordinate system. |
| N5 | Start position |
| N8 | Activation of coupled motion of rotatable tool axes C/C1. |
| N10 | C/C1 axis rotates to 225° stroke |
| N15 | C/C1 axis rotates to 45° stroke |

| | |
|---|---|
| N20 | C/C1 axis rotates to 90°, no stroke initiation |
| N25 | Path segmentation: four strokes are executed with tool rotated to 90°. |
| N30 | Position |
| N32 | Activate tangential control, offset angle of rotatable tool axes 180° |
| N35 | Path segmentation: three strokes with active tangential control and an offset angle of 180° |
| N40 | C–/C1 rotates to 225° (180° + 45° INC) tangential control deactivated because path is not segmented ↓ stroke |
| N42 | Tangential control without offset |
| N45 | Path segmentation: three strokes with active tangential control but without offset angle |
| N50 | Deactivation of stroke initiation + tangential control |

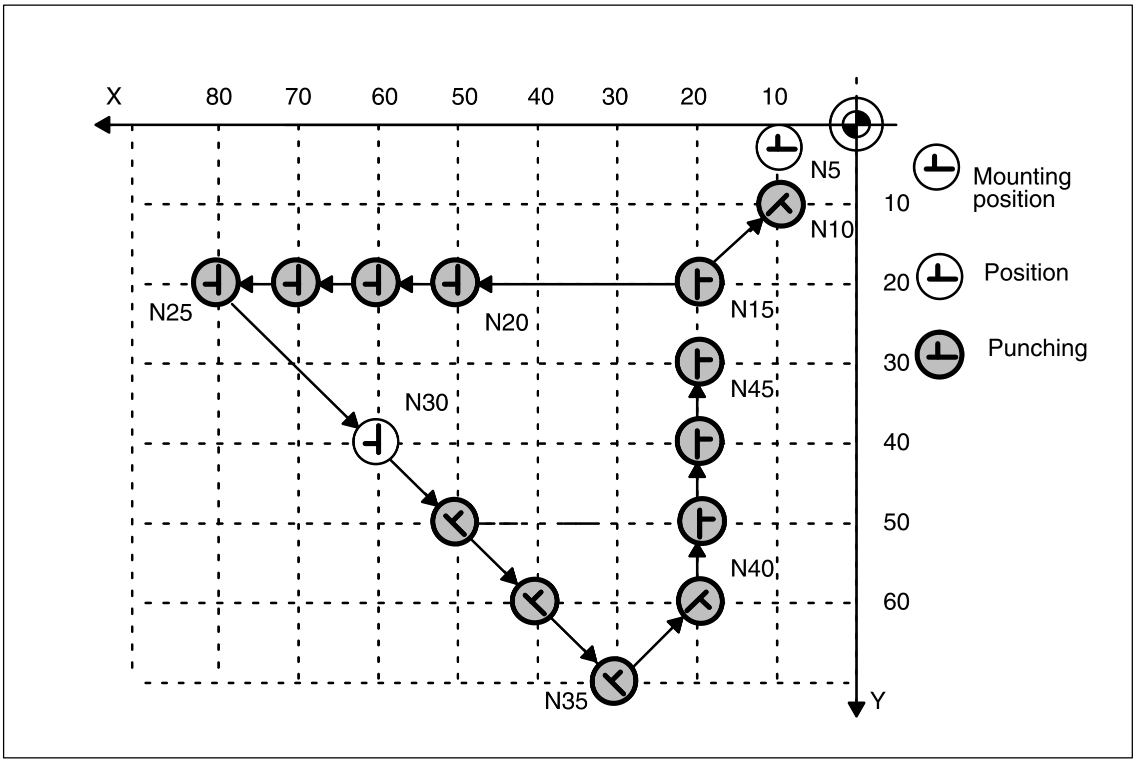



Fig. 2-11 Illustration of programming example in XY plane

**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.

```
:
:
N2    TANG (C, X, Y, 1, "B")
N5    G0    F60     X10     Y10
N8    TRAILON (C, C1, 1)
N9    TRAILON (C, –90)
N10   G02   X30     Y30     I20      J0     SPN=2   PON
N15   G0    X70     Y10     SPOF
N17   TANGON (C, 90)
N20   G03   X35,86  Y24,14  CR=20    SPP=16         SON
N25   G0    X74,14  Y35,86  C0       PON
N27   TANGON (C, 0)
N30   G03   X40     Y50     I–14,14  J14,14 SPN =5  SON
N35   G0    X30     Y65     C90      SPOF
N40   G91   X–10    Y–25    C180
N43   TANGOF
N45   G90   G02     Y60     I0       J10    SPP=2   PON
N50   SPOF
N55   M2
```

## Explanations

Table 2-2

| Set | Remarks |
|---|---|
| N2 | Definition of master and slave axes, C is slave axis for X and Y in the base coordinate system. |
| N5 | Start position |
| N8 | Activate coupled motion of rotatable tool axes C/C1 for punch and die. |
| N9 | Activate tangential control with offset 270°. |
| N10 | Circular interpolation with path segmentation, 2 strokes are executed with 270° offset angle and tangential alignment along circular path. |
| N15 | Positioning. |
| N17 | Activate tangential control with offset 90°. |
| N20 | Circular interpolation with path segmentation, four strokes are executed with 90° offset angle and tangential alignment along circular path. |
| N25 | Rotation of tool axes to 0°, stroke. |
| N27 | Activate tangential control with offset 0°. |
| N30 | Circular interpolation with path segmentation, five strokes with offset angle 0° and tangential alignment along circular path. |
| N32 | Activate tangential control, offset angle of rotatable tool axes 180° |
| N35 | Position without active tangential control. |
| N40 | Positioning, C axis rotates to 270° |
| N43 | Deactivate tangential control. |
| N45 | Circular interpolation with path segmentation, two strokes without tangential control where C=270°. |
| N50 | Punching OFF. |

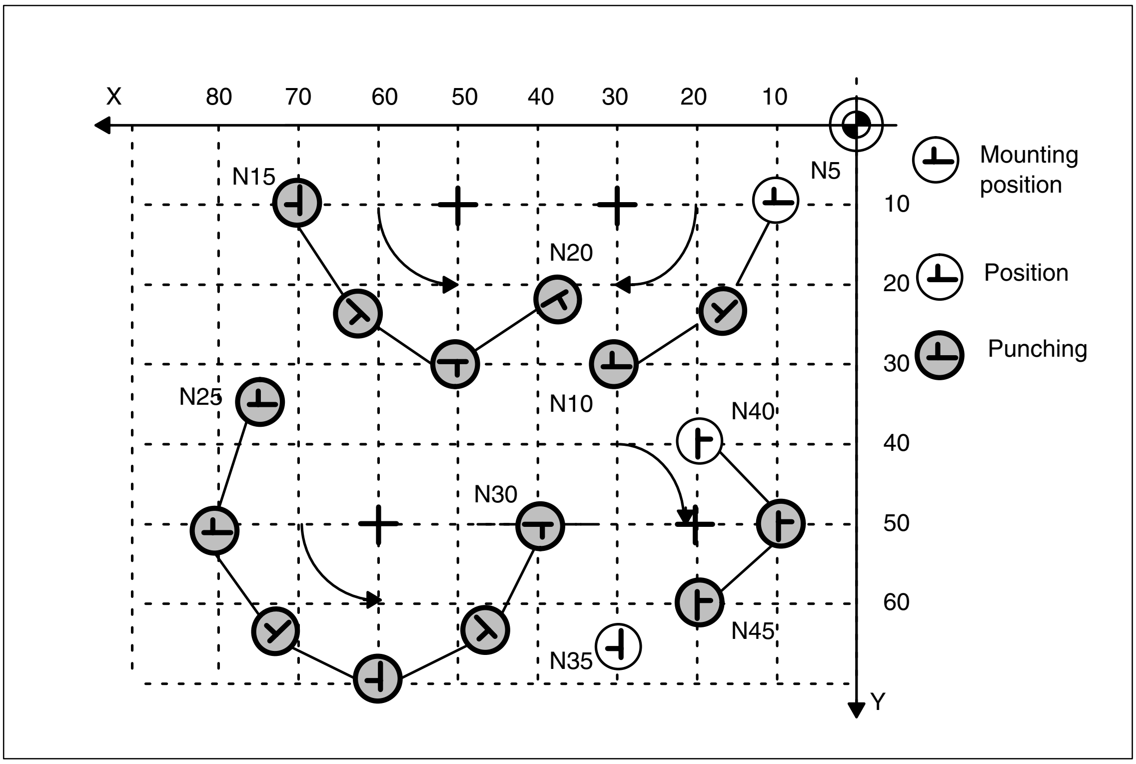



Fig. 2-12 Illustration of programming example in XY plane

## 2.5 Protection zones

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:** /FB/, A3, “Axis Monitoring Functions, Protection Zones”

■

# Supplementary Conditions 3

**Availability of function "Punching and nibbling"**

The function is an option and is available for

- SINUMERIK 840D with NCU 572 and 573, SW 3 and higher.

■

# Data Descriptions (MD, SD) 4

## 4.1 Channelspecific machine data

| | | | | |
|---|---|---|---|---|
| **26000**<br>MD number | **PUNCHNIB_ASSIGN_FASTIN**<br>Hardware assignment for input byte with stroke control | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: plus |
| Changes effective after POWER ON | | Protection level: 2 / 7 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 3.1 | |
| Meaning: | This data defines which input byte is to be used for the signal "Stroke active".<br>= 1: On-board inputs (4 high-speed NCK inputs) are used<br>= 2, 3, 4, 5 The external digital NCK inputs are used<br>**Example**:<br>Value "0000 0001" Stroke active is HIGH active<br>Value "0001 0001" Stroke active is LOW active<br><br>**Note:**<br>This MD is not compatible with earlier SW versions. The HIGH word now acts as an inversion mask. In earlier SW versions (< 3.2), the "Stroke active" signal was always LOW active. | | | |
| Related to .... | NIBBLE_PUNCH_INMASK[n] | | | |
| References | /FB/, A4, Digital and Analog NCK I/Os<br>The signal is high-active as standard with SW 3.2 and higher, i.e. open-circuit monitoring is implemented. If the signal needs to be low-active, the MD must be set, for example, to a value of "H 0001 0001" for the outboard inputs. | | | |

| **26002**<br>MD number | **PUNCHNIB_ASSIGN_FASTOUT**<br>Hardware assignment for output byte with stroke control | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: plus |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: DWORD | Applies from SW: 3 | |
| Meaning: | This data defines which output byte is to be used for the stroke control.<br>= 1: On-board outputs (4 high-speed NCK outputs) are used<br>= 2, 3, 4, 5 The external digital NCK outputs are used | |
| Related to .... | NIBBLE_PUNCH_OUTMASK[n] | |
| References | /FB/, A4, Digital and Analog NCK I/Os | |

| **26004**<br>MD number | **NIBBLE_PUNCH_OUTMASK[n]**<br>Screen form for high-speed output bits | |
|---|---|---|
| Default setting: see below | Minimum input limit: 0 | Maximum input limit: 128 |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: BYTE | Applies from SW: 3 | |
| Meaning: | A total of 8 byte screen forms for the output of high-speed bits can be defined with this data. Two of these are used at the current time.<br>The default setting for this data is as follows:<br>NIBBLE_PUNCH_OUTMASK[0] = 1: $2^0$ = first bit for the first punch interface (SPIF1)<br>NIBBLE_PUNCH_OUTMASK[1] = 0: Second punch interface (SPIF2), not available by default<br>NIBBLE_PUNCH_OUTMASK[2] = 0<br>...<br>NIBBLE_PUNCH_OUTMASK[7] = 0<br><br>Note:<br>The significance of the bit to be defined must be input. | |
| Application example(s) | NIBBLE_PUNCH_OUTMASK[0] = 1 —> The first bit (bit 0) is defined<br>NIBBLE_PUNCH_OUTMASK[0] = 4 —> The third bit (bit 2) is defined | |
| Special cases, errors, ...... | Only NIBBLE_PUNCH_OUTMASK[0] is relevant. This is used to define the output bit for the signal “Initiate stroke”. | |
| Related to .... | PUNCHNIB_ASSIGN_FASTOUT | |

| | Bit: 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Significance: | $2^7=128$ | $2^6=64$ | $2^5=32$ | $2^4=16$ | $2^3=8$ | $2^2=4$ | $2^1=2$ | $2^0=1$ |

Example:
“8”must be entered in the machine data in order to define bit 3.

| **26006**<br>MD number | **NIBBLE_PUNCH_INMASK[n]**<br>Screen form for high-speed input bits | |
|---|---|---|
| Default setting: see below | Minimum input limit: 0 | Maximum input limit: 128 |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: BYTE | Applies from SW: 3 | |

| **26006**<br>MD number | **NIBBLE_PUNCH_INMASK[n]**<br>Screen form for high-speed input bits |
|---|---|
| Meaning: | A total of 8 byte masks for the output of high-speed bits can be defined with this data.<br>The default setting for this data is as follows:<br>NIBBLE_PUNCH_INMASK[0]=1: $2^0$ = first bit for the first punch interface (SPIF1)<br>NIBBLE_PUNCH_INMASK[1]=4: Second punch interface (SPIF2), not available by default<br>NIBBLE_PUNCH_INMASK[2]=0<br>...<br>NIBBLE_PUNCH_INMASK[7]=0<br><br>Note:<br>The significance of the bit to be defined must be input (refer to MD 26004: NIBBLE_PUNCH_OUTMASK[n]). |
| Application example(s) | NIBBLE_PUNCH_INMASK[0] = 1 —> The first bit (bit 0) is defined<br>NIBBLE_PUNCH_INMASK[0] = 4 —> The third bit (bit 2) is defined |
| Special cases, errors, ...... | Only NIBBLE_PUNCH_INMASK[0] is relevant. This is used to define the input bit for the signal “Stroke active”. |
| Related to .... | PUNCHNIB_ASSIGN_FASTIN |

| **26010**<br>MD number | **PUNCHNIB_AXIS_MASK**<br>Definition of punching and nibbling axes | |
|---|---|---|
| Default setting: 7 | Minimum input limit: 0 | Maximum input limit: plus |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: DWORD | Applies from SW: 3 | |

| | |
|---|---|
| Meaning: | This data is used to define which channel axes shall be treated as punching/nibbling axes. This data setting determines above all the response of synchronous axes in stroke control mode and with automatic path segmentation.<br>When the axes are defined, the bit significance must be entered according to the following diagram:<br>Bit: 7 6 5 4 3 2 1 0<br>Significance: $2^7$=128 $2^6$=64 $2^5$=32 $2^4$=16 $2^3$=8 $2^2$=4 $2^1$=2 $2^0$=1<br>Example:<br>If the first two axes are to be defined as the punching/nibbling axes, then “3”must be entered (corresponds to setting of bits 1 and 0).<br>PUNCHNIB_AXIS_MASK = 3<br>The first two axes – typically x and y – are punching/nibbling axes.<br><br>PUNCH_NIB_AXIS_MASK = 11<br>The first two axes (typically x and y) and the third axis (e.g. an axis for the rotatable tool) are punching/nibbling axes. In this case, the response of the 3rd axis with automatic path segmentation can be defined via PUNCH_PARTITION_TYPE.<br><br>Independently of the above, this MD also defines whether the path of the A axis must be segmented by means of SPN=<value> if it is programmed without path axes.<br>In addition, this data also allows single path axes to be treated differently in stroke control mode and with path segmentation. To illustrate this option, let us assume that we have a machine with path axes x, y and rotary axis A. However, we do not want the Z axis traversing paths to be segmented into individual sections nor do we want a stroke to be initiated at the end of a block which contains only a Z axis movement.<br>It is possible to achieve the desired response by setting PUNCHNIB_AXIS_MASK = 11. In this case, pure Z movements are not taken into account by the path segmentation function. |
| Related to .... | PUNCH_PARTITION_TYPE |

| | | | | | |
|---|---|---|---|---|---|
| **26012**<br>MD number | **PUNCHNIB_ACTIVATION**<br>Activation of punching and nibbling functions | | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 3 | |
| Meaning: | This MD defines in what way punching and nibbling functions can be activated:<br>PUNCHNIB_ACTIVATION = 0<br>None of the punching or nibbling functions can be activated. The automatic path segmentation is the only exception if it is enabled via MD: PUNCH_PATH_SPLITTING.<br>PUNCHNIB_ACTIVATION = 1<br>The functions are activated via language commands. If M functions are to be used, then they must be programmed using macros.<br>PUNCHNIB_ACTIVATION = 2<br>The M functions are interpreted directly by the software. Language commands can still be used.<br>Note: This option is intended only as a temporary solution. | | | | |
| Related to .... | PUNCH_PATH_SPLITTING<br>NIBBLE_PUNCH_CODE[n] | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **26014**<br>MD number | **PUNCH_PATH_SPLITTING**<br>Activation of automatic path segmentation | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 3 | |
| Meaning: | This machine data defines whether it should be possible to activate the automatic path segmentation function, even if punching-specific functionality is not available.<br>PUNCH_PATH_SPLITTING = 0<br>Automatic path segmentation can only be activated if punching mode is active and is deactivated as soon as punching mode is deactivated.<br>PUNCH_PATH_SPLITTING = 1<br>In this case, automatic path segmentation can be activated for geometry axes even when punching mode is not active. | | | | |

| **26016**<br>MD number | **PUNCH_PARTITION_TYPE**<br>Behavior of single axes with active automatic path segmentation | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – |
| Data type: DWORD | Applies from SW: 3 | |
| Meaning: | This machine data determines how single axes, which are also nibbling axes according to MD: PUNCHNIB_AXIS_MASK, should respond. It is assumed that rotary axis A is defined as the 4th and nibbling axis in addition to the three path axes X, Y, Z. In this case, there are the following options for defining its response with automatic path segmentation and in stroke control mode:<br>PUNCH_PARTITION_TYPE = 0<br>No special response in the case of automatic path segmentation. If the single axis is programmed together with path axes in one block, then its total traversing path is segmented in accordance with the path axes, i.e. the purely geometric relationship between single axis and path axes is identical to that for non-segmented movements. If the single axis is programmed without path axes, but with SPN=<value>, then the path is segmented according to the programmed SPN value.<br>PUNCH_PARTITION_TYPE = 1<br>In this case, the path of the single axis is generally (i.e. regardless of the currently active interpolation mode) traversed in the first segment if the axis is programmed together with path axes.<br>PUNCH_PARTITION_TYPE = 2<br>In this case, the single axis responds to linear interpolation in the same way as described under<br>PUNCH_PARTITION_TYPE = 1,<br>and to all other interpolation modes as described under PUNCH_PARTITION_TYPE = 0. | |
| Related to .... | PUNCHNIB_AXIS_MASK | |

| **26018**<br>MD number | **NIBBLE_PRE_START_TIME**<br>Automatic prestart time | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: |
| Changes effective after POWER ON | Protection level: | Unit: – |
| Data type: | Applies from SW: 3.1 | |
| Meaning: | To minimize any dead times due to the reaction time of the punching unit, it is possible to release the stroke before reaching the in-position window of the axes. The reference time for this is the interpolation end. Since there is normally a delay of some interpolation cycles after reaching the interpolation end (depending on the machine dynamics) until the axes reach their true position, the prestart time is a delay time with respect to reaching the interpolation end. The function is therefore coupled to G603 (block change at the end of interpolation). The time can be set via the machine data NIBBLE_PRE_START_TIME. Example: With an interpolation cycle of 5msec, a stroke shall be released two cycles after reaching the interpolation end. In this case, the value 0.010 s must be selected for NIBBLE_PRE_START_TIME. If a value that is not integrally divisible by the set interpolation time is selected, then the stroke is initiated in the interpolation cycle following the set time. | |
| Related to .... | | |

| **26020**<br>MD number | **NIBBLE_SIGNAL_CHECK**<br>Monitoring of input signal | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: |
| Changes effective after POWER ON | Protection level: | Unit: sec |
| Data type: Flowld | Applies from SW: 3.1 | |
| Meaning: | If for example the stroke active signal is set by punch overshoot between the cycles, the interpolation is stopped. Furthermore, it is possible to generate the “Distorted punch signal” message as a function of machine data NIBBLE_SIGNAL_CHECK. | |
| Related to .... | | |

| | | | | | |
|---|---|---|---|---|---|
| **26008**<br>MD number | **NIBBLE_PUNCH_CODE[n]**<br>Definition of M functions (applies only to SW 3.1) | | | | |
| Default setting: see below | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 / 7 | | Unit: – |
| Data type: DWORD | | | | Applies only to SW: 3.1 | |
| Meaning: | This machine data defines the special M functions for punching and nibbling.<br><br>Default setting / Example<br>NIBBLE_PUNCH_CODE[0] = 0 / 20 End punching, nibble with M20<br>NIBBLE_PUNCH_CODE[1] = 23 / 23 End punching, nibble with M23<br>NIBBLE_PUNCH_CODE[2] = 22 / 22 Start nibbling<br>NIBBLE_PUNCH_CODE[3] = 25 / 25 Start punching<br>NIBBLE_PUNCH_CODE[4] = 26 / 26 Activation of dwell<br>NIBBLE_PUNCH_CODE[5] =122 / 122 Start nibbling with leader, stroke control on servo level<br>NIBBLE_PUNCH_CODE[6] =125 / 125 Start punching with leader, stroke control on servo level<br>NIBBLE_PUNCH_CODE[7] = 0 / 0 Not used (available soon) | | | | |
| Special cases, errors, ...... | If MD: PUNCHNIB_ACTIVATION = 2 (M functions are interpreted directly by software), then MD: NIBBLE_PUNCH_CODE[0] = 20 must be set. | | | | |
| Related to .... | PUNCHNIB_ACTIVATION | | | | |

## 4.2 Channelspecific setting data

| | | | | |
|---|---|---|---|---|
| **42400**<br>SD number | **PUNCH_DWELL_TIME**<br>Dwell | | | |
| Default setting: 1.0 | | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after: immediately | | | | Unit: s |
| Data type: DOUBLE | | Applies from SW: 3 | | |
| Meaning: | This machine data is used to set the dwell between reaching the position and initiating the stroke movement.<br>The set value is rounded to whole multiples of the interpolation clock cycle (i.e. the value set here may deviate slightly to the dwell actually applied). | | | |
| Related to .... | MD 10710: PROG_SD_RESET_SAVE_TAB | | | |

| | | | | |
|---|---|---|---|---|
| **42402**<br>SD number | **NIBPUNCH_PRE_START_TIME**<br>Pre-start time | | | |
| Default setting: 1.0 | | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after: immediately | | | | Unit: s |
| Data type: DOUBLE | | Applies from SW: 3 | | |
| Meaning: | This setting data has exactly the same effect as machine data NIBBLE_PRE_START_TIME. Its primary purpose is to allow the pre-start time to be altered from the NC program so that it can be adapted to different metal sheet sizes and thicknesses. However, setting data 42402 is effective only when the machine data has been set to zero. | | | |
| Related to .... | MD 26018: NIBBLE_PRESTART_TIME | | | |

| | | | | |
|---|---|---|---|---|
| **42404**<br>SD number | **MINTIME_BETWEEN_STROKES**<br>Minimum time interval between two consecutive strokes | | | |
| Default setting: 1.0 | | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after: immediately | | | | Unit: s |
| Data type: DOUBLE | | Applies from SW: 5.3 | | |
| Meaning: | If a punching dwell time (PDELAYON) is programmed in addition to the minimum interval, then the two times are applied additively.<br>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: MINTIME_BETWEEN_STROKES has elapsed. | | | |
| Related to .... | MD 26018: NIBBLE_PRESTART_TIME | | | |

■

# Notes

# Signal Descriptions 5

## 5.1 Signal overview

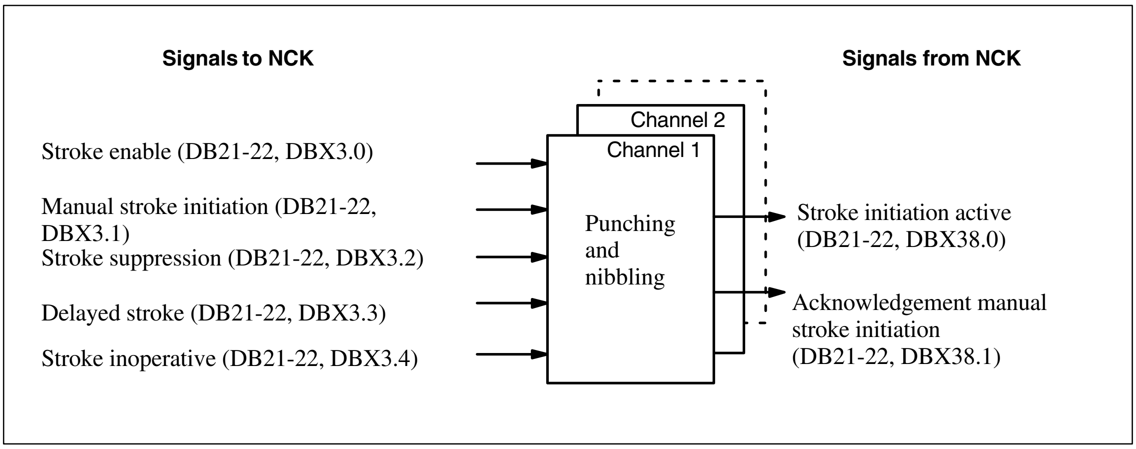



Fig. 5-1 PLC interface signals for “Punching and nibbling”

## 5.2 Signals to channel

| DB 21, 22<br>**DBX3.0**<br>Data Block | **No stroke enable**<br><br>Signal(s) to channel (PLC ——> NCK) | |
|---|---|---|
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 3 |
| Signal state 1 or signal transition 0 ——> 1 | This signal enables the punching strokes via the PLC.<br>1 signal: Stroke is disabled,<br>the NC must not enable punching strokes | |
| Signal state 0 or signal transition 1 ——> 0 | 0 signal: Stroke enable is present,<br>the NC may execute a punching stroke provided the enabling signal is not set. | |

| DB 21, 22<br>**DBX3.1**<br>Data Block | **Manual stroke initiation**<br><br>Signal(s) to channel (PLC ——> NCK) | |
|---|---|---|
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 3 |
| Signal state 1 or signal transition 0 ——> 1 | This signal permits a single stroke to be initiated in manual mode.<br>1 signal: Manual stroke is executed | |
| Signal state 0 or signal transition 1 ——> 0 | 0 signal: No effect | |

| DB 21, 22<br>**DBX3.2**<br>Data Block | **Stroke suppression**<br><br>Signal(s) to channel (PLC ——> NCK) | |
|---|---|---|
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 3 |
| Signal state 1 or signal transition 0 ——> 1 | This signal simply prevents execution of the stroke. The machine continues to operate. 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 by the path segmentation.<br>1 signal: Stroke suppression is active | |
| Signal state 0 or signal transition 1 ——> 0 | 0 signal: Stroke suppression is not active | |

| DB 21, 22<br>**DBX3.3**<br>Data Block | **Stroke inoperative**<br><br>Signal(s) to channel (PLC ——> NCK) | |
|---|---|---|
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 3 |
| Signal state 1 or signal transition 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.<br>In physical terms, the signal is identical to the signal “Stroke active” for the CNC, i.e. the system is wired such that the two signals are taken to the same NC input via an AND gate.<br>1 signal: Stroke inoperative (corresponds to signal “Stroke enable”) | |
| Signal state 0 or signal transition 1 ——> 0 | 0 signal: Stroke operative (corresponds to signal “Stroke enable”) | |

| **DB 21, 22**<br>**DBX3.4**<br>Data Block | **Delayed stroke**<br><br>Signal(s) to channel (PLC ––> NCK) | |
|---|---|---|
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 3 |
| Signal state 1 or signal transition 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.<br>1 signal: Delayed stroke is active | |
| Signal state 0 or signal transition 1 ––> 0 | 0 signal: Delayed stroke is not active | |

| **DB 21, 22**<br>**DBX3.5**<br>Data Block | **Manual stroke initiation**<br><br>Signal(s) to channel (PLC ––> NCK) | |
|---|---|---|
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 6.4 |
| Signal state 1 or signal transition 0 ––> 1 | This "manual stroke initiation" signal 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.<br>Successful stroke initiation is indicated to the PLC by the NCK –> PLC Interface signal "Manual stroke initiation acknowledgement" (DB21, ... DBX38.1).<br>1 signal: Manual stroke initiation is active | |
| Signal state 0 or signal transition 1 ––> 0 | 0 signal: Manual stroke initiation is not active | |

## 5.3 Signals from channel

| | | |
|---|---|---|
| **DB21, 22**<br>**DBX38.0**<br>Data Block | **Stroke initiation active**<br><br>Signal(s) from channel (NCK –> PLC) | |
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 3 |
| Signal state 1 or signal transition 0 ——> 1 | This signal indicates whether the stroke initiation is active.<br>1 signal: Stroke initiation is active | |
| Signal state 0 or signal transition 1 ——> 0 | 0 signal: Stroke initiation is not active | |

| | | |
|---|---|---|
| **DB21, 22**<br>**DBX38.1**<br>Data Block | **Acknowledgement of manual stroke initiation**<br><br>Signal(s) from channel (NCK –> PLC) | |
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 3 |
| Signal state 1 or signal transition 0 ——> 1 | This signal indicates whether a manual stroke has been initiated.<br>1 signal: Manual stroke has been initiated | |
| Signal state 0 or signal transition 1 ——> 0 | 0 signal: Manual stroke has not been initiated | |

■

# Examples 6

Examples of defined start of nibbling operation

**1. Example:**

```
:
:
N10   G0     X20    Y120   SPP= 20      Position 1 is approached
N20   X120   SON                        Defined start of nibbling,
                                        first stroke on "1", last
                                        stroke on "2"
N30   Y20                               Defined start of nibbling,
                                        first stroke on "3", last
                                        stroke on "4"
N40   X20                               Defined start of nibbling,
                                        first stroke on "5", last
                                        stroke on "6"
N50          SPOF
N60   M2
```

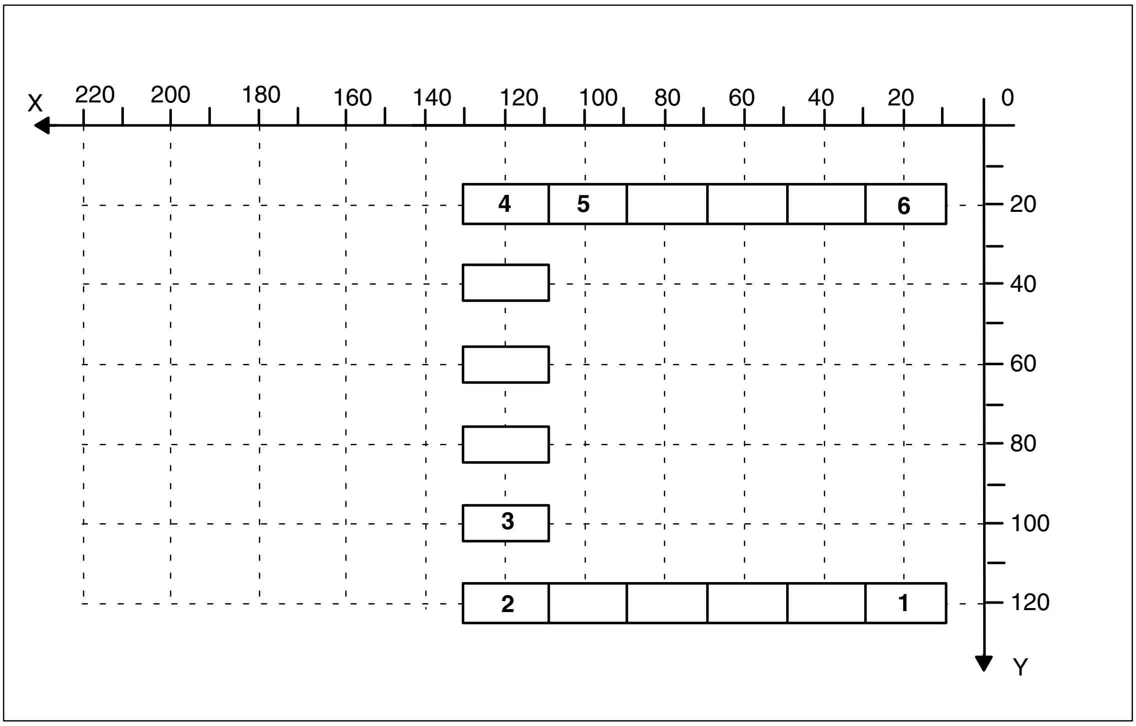



Fig. 6-1

**2. Example:**

This example utilizes the “Tangential control” function. Z has been selected as the name of the tangential axis.

```
:
:
N5    TANG  (Z, X, Y, 1, "B")           Definition of tangential axis
N8    TANGON      (Z, 0)                Selection of tangential control
N10   G0    X20   Y120                  Position 1 is approached
N20   X120  Z     SPN=20     SON        Defined start of nibbling,
                                        tangential control
                                        selected, first stroke on "1",
                                        last stroke on "2"
N30   SPOF  TANGOF                      Deselection of nibbling mode
                                        and deselection of tangential
                                        control
N38   TANGON      (Z, 0)                Selection of tangential control
N40   Y20   Z     SON                   Defined start of nibbling,
                                        tangential control selected,
                                        first stroke on "2", rotated
                                        through 90 degrees to
                                        block N20, last stroke on "3"
N50   SPOF
N60   M2
```

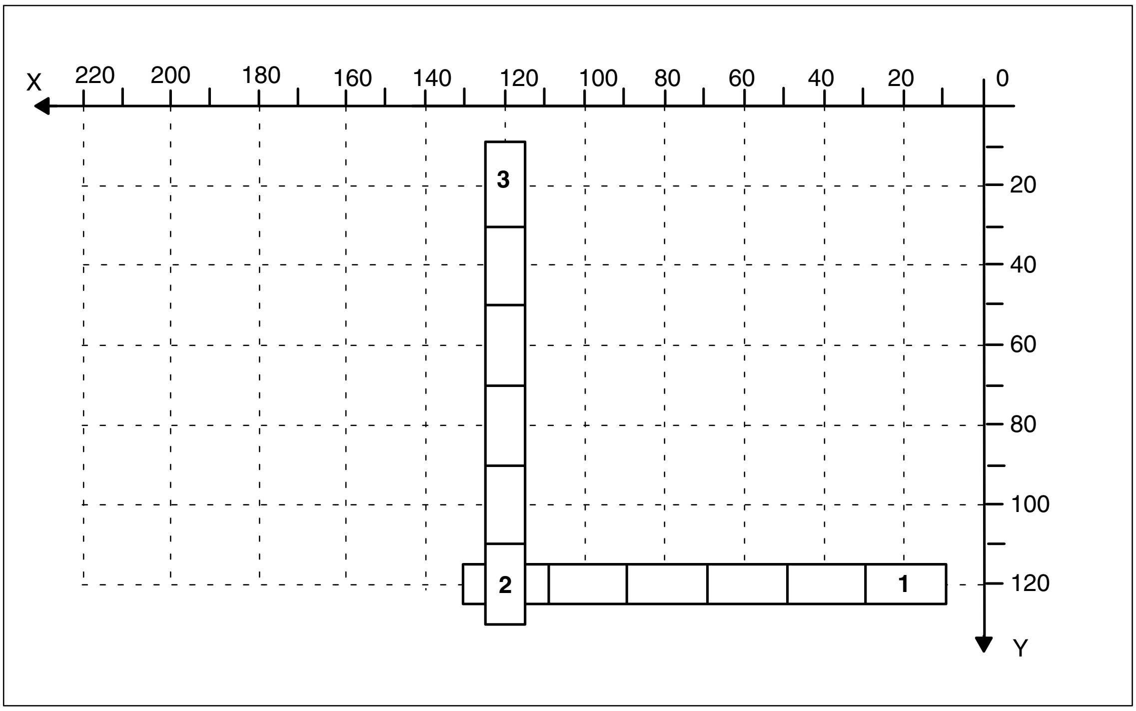



Fig. 6-2

(3) Example of **defined start of nibbling** illustrated in diagram below:

```
:
:
N5    G0    X10   Y10                      Position
N10   X90   SPN=20      SON                Defined start of nibbling,
                                           Five punching operations
N20   X10   Y30   SPP=1                    Initiation of punching
                                           operation at end of path
N30   X90                                  Four punching operations
                                           with SPP distance=20
N40   SPOF
N50   M2
```

(4) Example of **defined start of nibbling** illustrated in diagram below:

```
:
:
N5    G0    X10   Y10                      Position
N10   X90   SPN=4       SON                Defined start of nibbling,
                                           Five punching operations
N20   X10   Y30   PON                      Initiation of punching
                                           operation at end of path
N30   X90   SPN=4                          Four punching operations
N40   SPOF
N50   M2
```

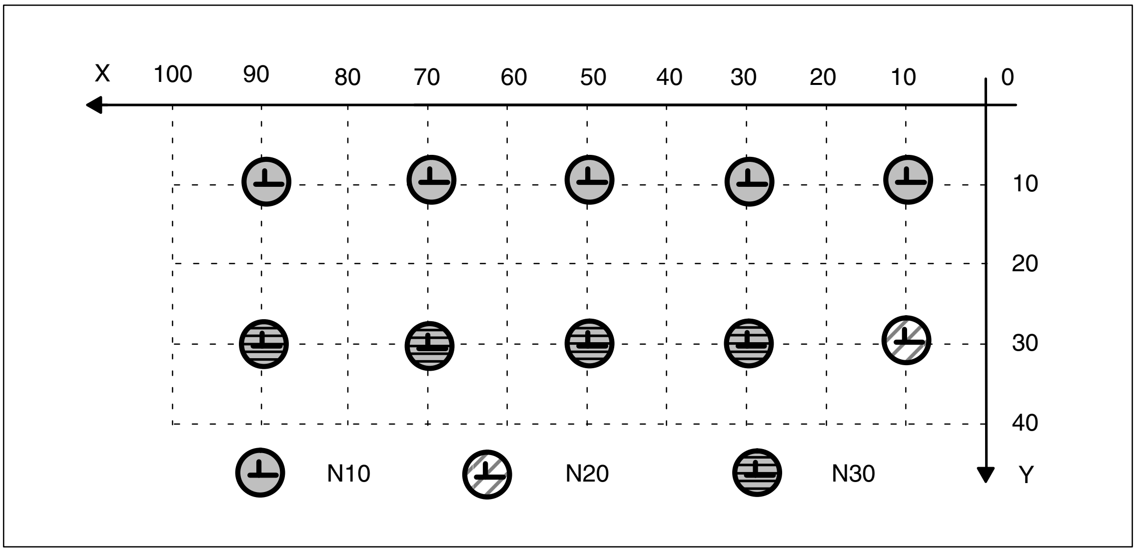



Fig. 6-3 Example of defined start of nibbling operation

(5) Examples of E programming **without defined start of nibbling** illustrated in diagram below:

:
:

| | | | | | |
|---|---|---|---|---|---|
| N5 | G0 | X10 | Y30 | | Position |
| N10 | X90 | SPN=20 | | PON | No defined start of nibbling, Four punching operations |
| N15 | Y10 | | | | Initiation of punching operation at end of path |
| N20 | X10 | SPP=20 | | | Four punching operations with distance E20 |
| N25 | SPOF | | | | |
| N30 | M2 | | | | |

(6) Examples of H programming **without defined start of nibbling** illustrated in diagram below:

:
:

| | | | | | |
|---|---|---|---|---|---|
| N5 | G0 | X10 | Y30 | | Position |
| N10 | X90 | SPN=4 | | PON | No defined start of nibbling, Four punching operations |
| N15 | Y10 | | | | Initiation of punching operation at end of path |
| N20 | X10 | SPN=4 | | | Four punching operations |
| N25 | SPOF | | | | |
| N30 | M2 | | | | |

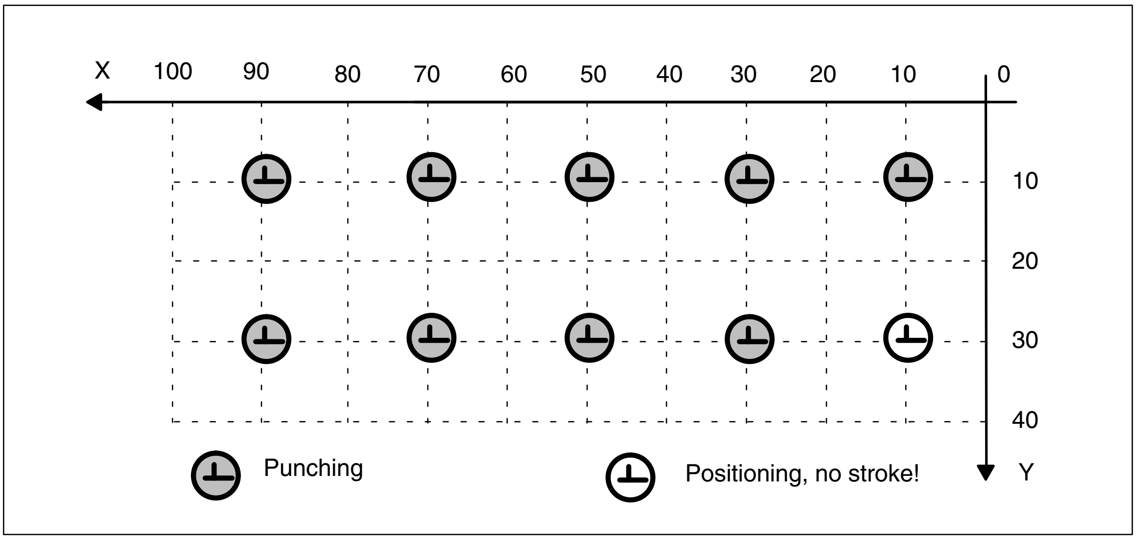



Fig. 6-4 Examples of E/H programming without defined start of nibbling

Application example: E value for punching

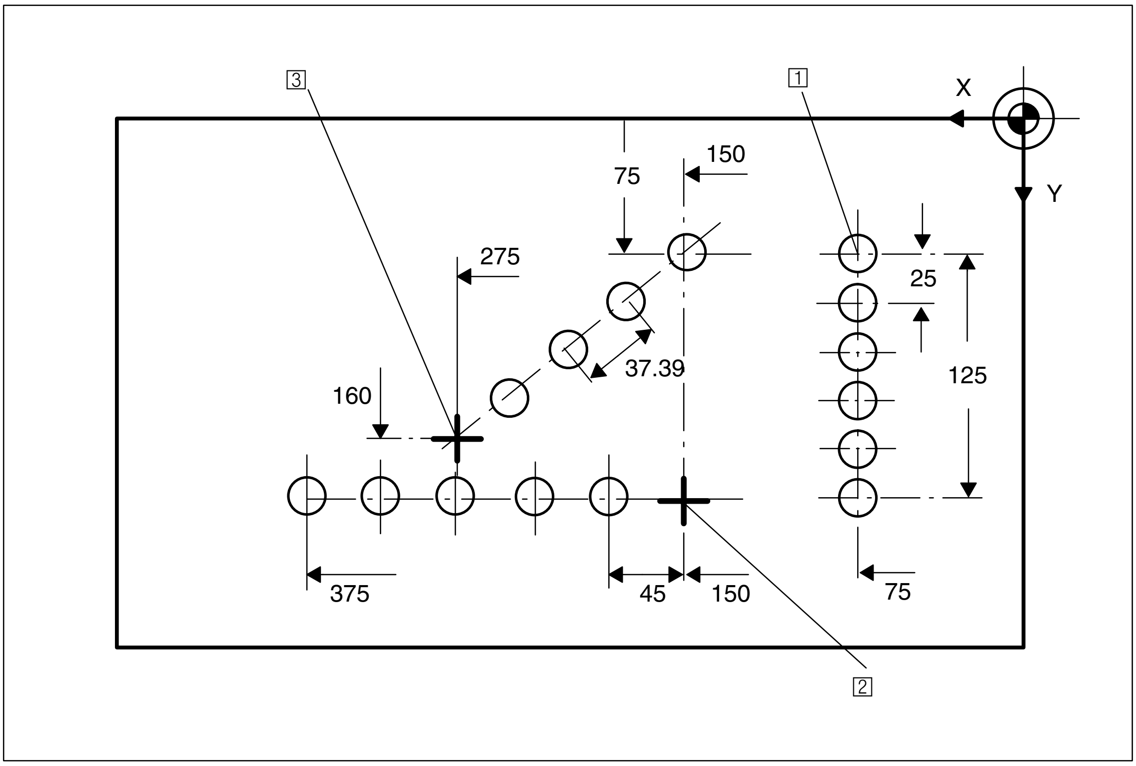



Fig. 6-5 Workpiece

Program extract:

| | | | | | | |
|---|---|---|---|---|---|---|
| N100 | G90 | X75 | Y75 | F60 | PON | Position at starting point 1 of vertical row of holes, punch single hole |
| N110 | G91 | Y125 | SPP=25 | PON | | End point coordinates (incremental), feedrate value: 25 mm, activate punching |
| N120 | G90 | X150 | SPOF | | | Absolute dimensioning, position at starting point 2 of horizontal row of holes |
| N130 | X375 | SPP=45 | PON | | | End point coordinates, feedrate value: 45 mm |
| N140 | X275 | Y160 | SPOF | | | Position at starting point 3 of inclined row of holes |
| N150 | X150 | Y75 | SPP=40 | PON | | End point coordinates, programmed feedrate value: 40 mm, calculated feedrate value: 37.79 mm |
| N160 | G00 | Y300 | SPOF | | | Position |

■

# Notes

# Data Fields, Lists 7

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **Channel-specific (signals to channel)** | | | |
| 21–22 | 3.0 | No stroke enable | |
| 21–22 | 3.1 | Manual stroke initiation | |
| 21–22 | 3.2 | Stroke suppression | |
| 21–22 | 3.3 | Stroke inoperative | |
| 21–22 | 3.4 | Delayed stroke | |
| 21–22 | 3.5 | Manual stroke initiation | |
| **Channel-specific (signals from channel)** | | | |
| 21–22 | 38.0 | Stroke initiation active | |
| 21–22 | 38.1 | Acknowledgement of manual stroke initiation | |

## 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channelspecific ($MC_ ... )** | | | |
| 20150 | GCODE_RESET_VALUES[n] | Reset G groups | /K1/ |
| 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] | Screen form for high-speed output bits | |
| 26006 | NIBBLE_PUNCH_INMASK[n] | Screen form for high-speed input bits | |
| 26008 | NIBBLE_PUNCH_CODE[n] | Definition of M functions (applies only to SW 3.1) | |
| 26010 | PUNCHNIB_AXIS_MASK | Definition of punching and nibbling axes | |
| 26012 | PUNCHNIB_ACTIVATION | Activation of punching and nibbling functions | |

| Number | Names | Name | Reference |
|---|---|---|---|
| 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 | |

## 7.3 Setting data

| Number | Names | Name | Reference |
|---|---|---|---|
| **Channelspecific ($SC_ ... )** | | | |
| 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 (SW 5.3 and higher) | |

## 7.4 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 |

## 7.5 Alarms

For detailed descriptions of the alarms, please refer to
**References:** /DA/, “Diagnostics Guide”
and the online help of MMC 101/102/103 systems.

■

## Notes

# SINUMERIK 840D/840Di/810D
# Description of Functions
# Extended Functions (FB2)

# Positioning Axes (P2)

■

# Brief Description 1

In addition to axes for machining, 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 quill/sleeve or steady

**Positioning axes**

The Positioning axes function allows axes for auxiliary motions to be integrated more easily into the control system.
The integration of the positioning axes is simpler

- During programming:
  The axes are programmed together with the axes for workpiece machining in the same parts program, without having to sacrifice valuable machining time.
- During program testing/start-up:
  Program testing and start-up is performed simultaneously for all axes.
- During operation:
  Operation and monitoring of the machining process commence simultaneously for all axes.
- During PLC configuring/start-up:
  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 configuration:
  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:

- With a path interpolator:
  Linear interpolation (G01), circular interpolation (G02/G03), spline interpolation, etc.
- With an axis interpolator:
  Each channel has an axis interpolator. When a positioning axis is programmed, an axis interpolator is started in the control (with linear interpolation G01).
- End of motion criterion (SW 6.1 and higher):
  The programmed end position of a positioning axis is reached when the end of motion criterion FINEA, COARSA, IPOENDA is fulfilled.
- Path axes in a rapid traverse motion (SW 6.1 and higher):
  Path axes can be moved in rapid traverse (G0) either via linear interpolation or non-linear interpolation.
- Autonomous single-axis operations (SW 6.3 and higher):
  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 so as to transfer control of a specific axis/spindle to the PLC.
- Control by the PLC:
  All channel-specific signals act to the same extent on path and positioning axes. Exceptions to this are explained in Chapter 2.

  Positioning axes can be controlled via additional, axis-specific signals. These signals are described in Chapter 2.

  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.

■

# Detailed Description 2

In addition to the axes required for machining, a complex modern machine tool can be equipped with further axes for auxiliary movements. The axes for machining a workpiece are known as path axes: Within the channel they are guided by the interpolator such that they start simultaneously, accelerate, reach the end point and stop together.
The auxiliary axes include:

- Axis for tool magazine
- Axis for workpiece transport
- Axis for pallet transport
- Axis for loader
- Axis for tool changer
- Axis for quill/sleeve or steady

Many of these axes were previously manipulated hydraulically and triggered by the part program by means of an auxiliary function.
With control of the axis in the NC, the axis can be addressed by name in the part program and the actual position displayed on the screen.

Positioning axes are traversed independently of the path axes with their own dedicated axis-specific feedrate.
Synchronous axes and geometry axes can be traversed non-modally as positioning axes.
Special travel instructions are provided for positioning axes
POS[...], POSA[...]

## 2.1 Selection of positioning axes

When axes are provided for auxiliary movements on a machine tool, the required properties will decide whether the axis is to be:

- programmed in a separate parts program
  —> see Subsection 2.1.1 “Separate channels”
- programmed in the same parts program as the machining process
  —> see Subsection 2.1.2 “Positioning axes”
- triggered exclusively from the PLC during machining
  —> see Subsection 2.1.3 “Concurrent positioning axes”

### 2.1.1 Separate channel

A channel represents a self-contained NC which, with the aid of a parts program, can be used to control the movement of axes, spindles and machine functions independently of other channels.

Independence between channels is assured by means of the following provisions:

- An active part program per channel
- Channel-specific signals such as
  - IS “NC Start” (DB21, ... DBX7.1)
  - IS “NC stop” interface signal (DB21, ... DBX7.3)
  - IS “Reset” (DB21, ... DBX7.7)
- 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 modification of programs:
  - Single block
  - Dry run (DRY RUN)
  - Block search
  - Program test

Please see the following for more information on channel functionality:
**References:** /FB/, K1, “Mode Groups, Channels, Program Operation Mode”

## 2.1.2 Position axis

Positioning axes are programmed together with path axes, i.e. with the axes that are responsible for workpiece machining.
Instructions for both 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 2.2).
The block change depends on the type of positioning axis programmed (see Section 2.3).

- Type 1 The block change occurs when all path and positioning axes have reached their programmed end point.
- Type 2 The block change occurs when all path axes have reached their programmed end point.
- Type 3 With SW 6.2 and higher,
  It is possible to set the block change within the braking ramp of the single axis interpolation if the conditions for the motion end and the block change are fulfilled for the path interpolation.

With positioning axes of type 2, it is possible to approach the programmed end position across several block boundaries.

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).

**SW 6.2 and higher**

**Single-axis interpolation and path axes with G0 as positioning axis**

For single-axis interpolation, it is also possible to set a new motion end condition for the block change in the braking ramp. In addition, each path axis can be traversed as positioning axis in rapid traverse movement (G0). Thus all axes travel to their endpoint 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.

**Independence of path and positioning axes**

The mutual independence of path and positioning 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 “axisspecific delete distance-to-go” interface signal.

**Dependence of positioning axes**

Positioning axes are dependent in the following respects:

- A shared parts program
- Starting of positioning axes only at block boundaries in the parts program
- New end of motion condition for block change can now be programmed in the braking ramp of single-axis interpolation (SW 6.2 and higher).
- With rapid traverse movement G0, path axes traverse as positioning axes - one of two different modes selectable (SW 6 and higher).
- No rapid traverse override
- The signals
  - IS “NC Start” (DB21, ... DBX7.1)
  - IS “NC stop” interface signal (DB21, ... DBX7.3)
  - IS “Reset” (DB21, ... DBX7.7)
  - IS “Read-in disable” (DB21, ... DBX6.1)

  act on the entire channel and therefore on positioning axes.
- Program-specific and channel-specific alarms 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
- In SW 6 and higher, the last block processed in the search run with a motion end condition serves as a container for setting all axes.
- Group 1 (modal movement commands) of the G functions (G0, G1, G2, etc.) does not apply to positioning axes.
  **References:** /PA/, “Programming Guide”

**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

SW 6.2 and higher offers the following enhancements:

- 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.

Positioning axes are not suitable for multi-axis loaders that require interpolation between the axes (path interpolator).

### 2.1.3 Concurrent positioning axis

Concurrent positioning axes are positioning axes with the following attributes:

- 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 parts program is already being processed in the channel).
- Program command "WAITP" is required to move a concurrent positioning axis from the parts program immediately after power ON.
- The parts program continues to run uninhibited, even if the concurrent positioning axis has not reached the position defined by the PLC.
- SW 4.3 and higher
  Depending on the machine data AUTO_GET_TYPE, it is possible to traverse, from the parts program, using the programming command
  – "GET(axis)" or "WAITP(axis)" a concurrent positioning axis as a channel axis again, or with
  – "RELEASE (axis)" or "WAITP(axis)" a channel axis as a concurrent positioning axis.

**Activation from PLC**

The concurrent positioning axis is activated via FC 15 or FC 16 from the PLC.

- Feedrate (with feedrate setting = 0, the feed set in MD 32060: POS_AX_VELO (reset for positioning axis velocity) is applied.
- Absolute coordinates (G90), incremental coordinates (G91), absolute coordinates 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

**Applications**

Typical applications for concurrent positioning axes include:

- Tool magazines with manual loading and unloading during machining
- Tool magazines with tool preparation during machining

## 2.2 Motion behavior and interpolation functions

**Path interpolator**

Every channel has a path interpolator for a wide range of interpolation modes such as linear interpolation (G01), circular interpolation (G02/G03), spline interpolation, etc.

**Axis interpolator**

In addition to the path interpolator, each channel has the equivalent number of axis interpolators up to the maximum number of available channel axes. When a positioning axis is programmed, an axis interpolator is started in the control (with linear interpolation G01).

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.

In SW 5 and lower, the programmed end position of a positioning axis had been reached when the axis has reached the exact stop fine window (G09). Continuous-path mode (G64) was not possible with positioning axes.

In SW 5 and higher, the programmed end position of a positioning axis has been reached when the motion end condition FINEA, COARSA or IPOENDA is fulfilled.

### 2.2.1 Path axes traverse as positioning axes with G0 (SW 6.1 and higher)

**Functionality**

**Two optional modes for path axes at rapid traverse movement**

With rapid traverse movement (G0), path axes can optionally be moved in two different modes:

- **Linear interpolation**: The path axes are interpolated together.
  The tool movement programmed with G0 is executed at the greatest possible traversing velocity (rapid traverse). The rapid traverse velocity is defined for each axis separately in machine data MD 32000: 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. This corresponds to the previous functionality.
- **Non-linear interpolation**: Each path axis interpolates as a single axis (positioning axis) independently of the other axes at the rapid traverse velocity defined in MD 32000: MAX_AX_VELO associated with the axis. 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 both mode types, rapid override is channel-specific.

**Linear interpolation**

Linear interpolation is always performed in the following cases:

- With a G code combination with G0
  that does **not** allow a positioning axis movement, e.g. G40, G41, G42, G96, G961 and MD 20750: ALLOW_G0_IN_G96 == FALSE,
- with a combination of G0 and G64,
- when a compressor or transformation is active,
- in PTP (point-to-point) travel mode,
- when a contour handwheel is selected (FD=0),
- with an active frame and rotation of geometry axes,
- when nibbling is active for geometry axes.

**Non-linear interpolation**

With non-linear interpolation, the setting for the appropriate positioning axis BRISKA, SOFTA, DRIVEA or

MD 32420: JOG_AND_POS_JERK_ENABLE and
MD 32430: 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**

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.

---

**MD 20730 and MD 20732**

**Activation of non-linear interpolation: RTLIOF**

Definition of RTLI: **R**apid **T**raverse **L**inear **I**nterpolation

If the new machine data
MD 20730: G0_LINEAR_MODE == FALSE (Siemens mode) or
MD 20732: EXTERN_G0_LINEAR_MODE == FALSE (ISO mode) is activated, at rapid traverse (G0) the path axes are traversed as positioning axes.

If the new machine data
MD 20730: G0_LINEAR_MODE == TRUE (Siemens mode) or
MD 20732: EXTERN_G0_LINEAR_MODE == TRUE (ISO mode) is activated, linear interpolation can be deactivated with the NC parts program command RTLIOF and non-linear interpolation can be activated.

**Changeover to linear interpolation: RTLION**
RTLION switches back to linear interpolation.

The behavior of the path axes with G0 can be scanned with system variable $AA_G0MODE.

For an example of non-linear interpolation, see **Chapter 6**

### 2.2.2 Autonomous single-axis operations (SW 6.3 and later)

**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

**Supplementary 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.

**Sequence coordinator**

The sequence of autonomous single axis functions with the respective transfers is represented in a so-called **"Use Case"** overview:

| | |
|---|---|
| NCK has control | PLC wants to assume control of the axis/spindle |
| Use Case 1 | Cancel operation of axis/spindle |
| Use Case 2 | Stop axis/spindle |
| Use Case 3 | Resume axis/spindle motion |
| Use Case 4 | Reset axis/spindle |
| PLC has control | PLC wants to relinquish control of the axis/spindle |

**Axis control by PLC**

**Accept control of axis/spindle**

The channel behavior triggered by the NC program run is decoupled. Control of the axis/spindle is transferred via the axial VDI interface with IS "PLC controls axis" (DB31, ... DBX28.7)

**Description of operational sequence:**

1. PLC to NCK with IS "PLC controls axis" (DB31, ... DBX28.7) == 1
   Accept control of an axis
2. NCK checks: Is the axis a main run axis or a neutral axis
3. NCK reads MD 10008: MAXNUM_PLC_CNTRL_AXES
   to check whether the PLC can control another axis
4. NCK confirms transfer of axis control and passes the axis status to the PLC with
   IS "PLC controls axis" (DB31, ... DBX63.1) == 1
   System variable $AA_SNGLAX_STAT as scannable, current axis status.
5. PLC has accepted control of the axis/spindle

**Alternatives**

The channel status is “interrupted” because a channel stop signal is active. The axis is handled analogously to the sequence description. The following two alternatives are possible depending on the status of the axis to be controlled:

- The axis to be controlled by the PLC is **not** active.
  The stop status is also canceled. A subsequent axis start command results directly in an axis motion.
- The axis to be controlled by the PLC is not active.
  The stop status is not canceled. Create the axis status according to use case 2 “Stop axis”. Resume the axis motion according to use case 3 “Continue axis motion”.
- The channel executes an NC reset.
  This operation is asynchronous to the takeover of control by the PLC. The two previously mentioned alternatives can occur or the axis is assigned to the channel and is reset.

**Supplementary conditions**

The NCK must have booted.

Axes/spindles currently operating according to the NC program cannot be controlled by the PLC.

The NCK does not confirm acceptance of a program axis and outputs alarm 26072 “Channel %1 axis %2 cannot be controlled by PLC”.

If the value set in MD 10008: MAXNUM_PLC_CNTRL_AXES is exceeded, then axis control cannot be transferred to the PLC. Alarm 26070 “Channel %1 axis %2 cannot be controlled by PLC, max. no. exceeded” is generated.

**Relinquish axis control by PLC**

### Relinquish control of axis/spindle

Control of the axis/spindle is transferred via the axial VDI interface with IS “PLC controls axis” (DB31, ... DBX28.7)

**Description of operational sequence:**

1. PLC to NCK with IS “PLC controls axis” (DB31, ... DBX28.7) == 0 Relinquish control of an axis
2. NCK checks whether an axial alarm is active
3. NCK checks whether a movement has been activated that is not yet ended and stops this movement with an axial stop as per use case 2 “Stop axis/spindle”.
4. NCK executes an axial reset in accordance with use case 4 “Reset axis/spindle” by reading in and activating the requisite reset machine data for a single axis.
5. NCK confirms acceptance of axis control and passes the axis status to the PLC via axial VDI interface using
   IS “PLC controls axis” (DB31, ... DBX63.1) == 0
   IS “AxStop active” (DB31, ... DBB63.2) == 0
   IS “AxReset done” (DB31, ... DBB63.0) == 0
   Axis status with system variable $AA_SNGLAX_STAT == 0 “active”
6. The channel now has control over the axis/spindle.

**Alternatives**

The NCK confirms the transfer and internally sets the “stopped” channel status for the axis/spindle if it detects that

- the axis-control channel is in the “interrupted” status owing to a stop signal,
- a stop alarm is active for the channel or
- a stop alarm is active for the mode group.

**Supplementary 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. If an axial alarm is active, alarm 26074 “Channel %1 Deactivation of PLC control of axis %2 in current status not permitted” is generated.

**Applications**

An axis/spindle interpolated by the main run can be controlled by the PLC independently of the NC program via the following VDI interface signals:

- Cancel operation — IS “Delete distance-to-go” (DB31, ... DBX6.2)
- Stop axis/spindle — IS “AxStop, stop” (DB31, ... DBX28.6)
- Resume axis/spindle motion — IS “AxResume” (DB31, ... DBX28.2)
- Reset axis/spindle — IS “AxReset” (DB31, ... DBX28.1)

---

**Note**

The axis/spindle must be operating under PLC control. The condition basically applies to all applications: Use cases 1 to 4.

The exchange of signals at the VDI interface during autonomous single operations is described by means of machine axis examples in a comparison of PLC actions as the NCK reaction in Section 6.2.

For further information about the channel-specific VDI signal, please see:
**References:** /FB/, A2, “Various Interface Signals”

---

**Alarm handling for single axes**

Alarms with an axis parameter are only displayed and not handled as single axis alarms. It is sufficient to acknowledge these so-called “show alarms” with CANCEL.

---

**Note**

Only alarms that have to be cleared again with AxReset take effect on single axes.

---

**Use Case 1**

**Cancel operation of axis/spindle**

The Cancel Axis/Spindle function corresponds functionally to the channel-specific IS “Delete distance to go” (DB31, ... DBX6.2).

**Use Case 2**

### Stop axis/spindle

All axis motions controlled by the main run are stopped, thereby interrupting the following movements:

– Axis/spindle: PLC axis, asynchronous reciprocating axis or
command axis started via static synchronized action.
A following axis motion by the axis/spindle is **not** stopped.

– Overlaid motions such as:
$AA_OFF,
DRF handwheel traversal or online tool offset and
external zero offset.

**Description of operational sequence:**

1. PLC requests the NCK to stop the relevant axis with IS “AxStop, stop” (DB31, ... DBX28.6) == 1.
2. NCK brakes the axis along a ramp.
3. NCK switches the axis to the stop state and notifies the PLC of this status change via the VDI interface (NCK→PLC) as follows with:
   IS “AxStop active” (DB31, ... DBX63.2) == 0,
   IS “Travel command plus” (DB31, ... DBX64.7) == 0 or
   IS “Travel command minus” (DB31, ... DBX64.6) == 0 and
   IS “Exact stop fine” (DB31, ... DBX60.7) == 1 or
   IS “Exact stop coarse” (DB31, ... DBX60.6) == 1
   Axis status with system variable $AA_SNGLAX_STAT == 3 “interrupted”.
4. NCK ends this operation

**Alternatives**

The following scenarios for stopping the axis/spindle are possible:

1. Transfer and stop the axis/spindle simultaneously.
   When PLC informs NCK: PLC wants to accept the axis/spindle and it is simultaneously detected that the axis/spindle is to be stopped, then: NCK flags the axis/spindle as accepted and continues at the position where the interruption occurred.
2. The axis/spindle is executing several operating sequences:
   Stop all operating sequences apart from movements resulting from following axis motions. Following axis motion can only be stopped by stopping the leading axis!
3. PLC requests the NCK to stop the axis/spindle, an axial stop alarm for this axis is generated simultaneously:

   NCK decelerates the axis down a ramp and confirms the braking operation via OPI.
   At the same time, an alarm is signaled to the PLC with
   IS “AxAlarm” (DB31, ... DBX61.1) == 1 and the status system variable $AA_SNGLAX_STAT == 5.
   NCK switches the axis to the stop state and notifies the PLC of the change in axis status.
4. PLC requests the NCK to stop the axis/spindle, a stop alarm is generated simultaneously and the NC program is activated. The stop alarm is ignored and does not take effect.
5. A retraction movement of “Extended stop and retract” cannot be stopped.
   This retraction motion cannot be halted with IS “AxStop, stop” (DB31, ... DBX28.6) on the PLC interface.

**Supplementary conditions**

The PLC must already have accepted control of the axis/spindle. IS “AxStop, stop” (DB31, ... DBX28.6) is otherwise ignored.

**Use Case 3**

## Resume axis/spindle motion

The axis/spindle motions controlled by the main run and interrupted according to use case 2 “Stop axis” are resumed.

**Description of operational sequence:**

1. PLC requests the NCK to resume motion on the relevant axis with IS “AxResume” (DB31, ... DBX28.2) == 1.
2. NCK checks whether an axis/spindle axial alarm with the cancel criterion CANCELCLEAR or NCSTARTCLEAR is active and deletes it.
3. NCK checks whether the axis motion can be resumed – no interlocking due to an alarm – and switches it to the active state.
4. The axis motion is resumed and the PLC is notified of the change in status via the VDI interface (NCK→PLC) as follows:
   IS “AxStop active” (DB31, ... DBX63.2) == 0,
   IS “Travel command plus” (DB31, ... DBX64.7) == 1 or
   IS “Travel command minus” (DB31, ... DBX64.6) == 1 and
   IS “Exact stop fine” (DB31, ... DBX60.7) == 0 or
   IS “Exact stop coarse” (DB31, ... DBX60.6) == 0
   Axis status with system variable $AA_SNGLAX_STAT == 4 “active”
5. NCK ends this operation

**Alternatives**

The axis/spindle is executing several operating sequences:

- Resume all operating sequences.
- Following motions are dependent on the leading axis motion.

**Supplementary conditions**

The NCK detects the following fault scenarios:

- When the PLC prompts the axes/spindles to continue moving and this axis/spindle has not been accepted by the PLC, the IS “AxResume” (DB31, ... DBX28.2) is ignored.
- If the axis/spindle is not in a stopped state, the IS “AxResume” (DB31, ... DBX28.2) is ignored.
- If the motion of the axis/spindle must not be continued due to an active alarm, the IS “AxResume” (DB31, ... DBX28.2) is ignored.

**Use Case 4**

**Reset axis/spindle**

An axis/spindle is reset to its initial state.

**Description of operational sequence:**

1. PLC requests the NCK to reset the relevant axis with IS “AxReset” (DB31, ... DBX28.1) == 1.
2. NCK detects that the axis/spindle is active and switches it to the stopped state.
3. The interrupted sequences are canceled and notified to the PLC as for “Delete distance to go”.
4. The internal NCK states for the axis/spindle are reset.
5. The axial machine data effective on RESET become operative.

   **Note:** In the case of a channel reset, no axial machine data becomes operative for an axis controlled by the PLC.
6. If an axial reset has been executed, the following signals are set at the interface
   IS “AxReset done” (DB31, ... DBB63.0) == 1,
   IS “AxStop active” (DB31, ... DBX63.2) == 0
   and system variable $AA_SNGLAX_STAT == 1 “single axis in reset active”.
7. NCK ends this operation.

**Alternatives**

The NCK detects that the relevant axis/spindle is

- in the “stopped” state — Point 3 onwards of operational sequence description
- in the “StopByAlarm” state — Point 3 onwards of operational sequence description
- not active — Point 4 onwards of operational sequence description

**Supplementary conditions**

The PLC must already have accepted control of the axis/spindle. IS “AxReset” (DB31, ... DBX28.1) is otherwise ignored.

### 2.2.3 Autonomous single axis operations with numerically controlled ESR (SW 6.4 and higher)

**Extended stop numerically controlled**

With SW 6.4 and higher, 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
MD 37510: AX_ESR_DELAY_TIME1

ESR time for interpolatory braking of the single axis with
MD 37511: AX_ESR_DELAY_TIME2

The values of these axial machine data are applied, however, only if the axis/spindle is a single axis.
Numerically controlled extended stop and retraction is initiated by axial trigger $AA_ESR_TRIGGER[axis]. It works analogously to $AC_ESR_TRIGGER and applies exclusively to single axes.

**References:** /FB3/, M3, “Coupled axes and ESR”

**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) for type=1 or type=2 only
  POLFA(axis, , value, axis, type, ,axis, type).

---

**Note**

**Numerically 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.

**Numerically controlled extended retraction for single axes:**
The channel-specific numerically controlled extended retract function does **not** work on 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, such as
MD 37500: ESR_REACTION, $AA_ESR_ENABLE of the axis, etc., are set.

---

**Examples**

Extended **stopping** of a single axis:

```
MD 37500: ESR_REACTION[AX1]=22
MD 37510: AX_ESR_DELAY_TIME1[AX1]=0.3
MD 37511: AX_ESR_DELAY_TIME2[AX1]=0.06
 ...
$AA_ESR_ENABLE[AX1] = 1
$AA_ESR_TRIGGER[AX1]=1        ; the stop function starts working at this
                              ; point
```

Extended **retraction** of a single axis:

```
MD 37500: ESR_REACTION[AX1]=21
...
$AA_ESR_ENABLE[AX1] = 1
POLFA(AX1, 1, 20.0)           ; AX1 becomes the axial retraction position
                              ; 20.0 assigned (absolutely)
$AA_ESR_TRIGGER[AX1]=1        ; the retraction of AX1 starts at this point

POLFA(axis, type)             ; legal shortform for programming
POLFA(axis, 0/1/2)            ; high-speed deactivation / activation
```

---

**Warning**

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.

In particular after:
A power on, the retraction path or the retraction position must be reset.

```
POLFA(axis, 1, $AA_POLFA[Achse])   ; causes a preprocessing stop
POLFA(axis, 1)                     ; does not cause a preprocessing stop
```

---

## 2.3 Block change

Positioning axes can be programmed in the NC block individually or in combination with path axes.

Path axes and positioning axes are always interpolated separately (path interpolator and axis interpolators) and this causes them to reach their programmed end positions at different times.

There are two types of positioning axis, whose response differs with respect to block change:

- Type 1 The block change occurs when all path and positioning axes have reached their programmed end point.
- Type 2 Block change, if all path axes have reached their programmed end points according to G601, G602, G603.
- Type 3 With SW 6.2 and higher,
  it is possible to set the block change within the braking ramp of the single axis interpolation if the conditions for the motion end and the block change are fulfilled for the path interpolation.

**Positioning axis type 1**

### Block change at the programmed end point of all path axes and positioning axes

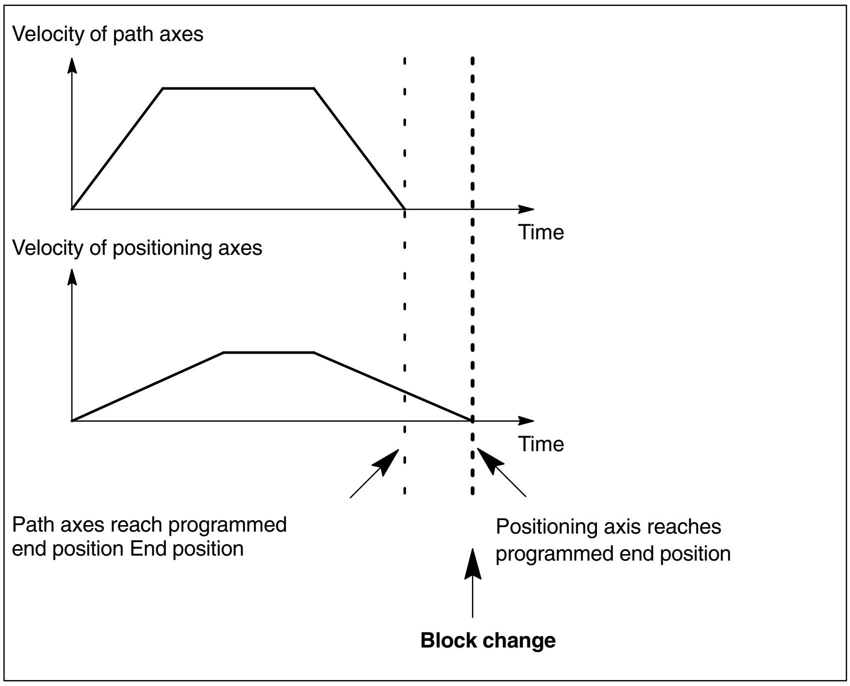



Fig. 2-1 Block change with positioning axis type 1, example of sequence

**Properties of type 1 positioning axis**

With SW 5 and lower, type 1 positioning axes have the following behavior:

- The block change occurs (NC block finished) when all the path and positioning axes have reached the respective motion end condition.
- Continuous-path mode (G64) is only possible for path axes if the positioning axes reach their motion end condition ahead of the path axes (this is not the case in the example in Fig. 2-1).
- Programming with
  POS[name] = end point FA[name] = feed
  or abbreviated with
  POSA[name] = end point
  in which case the feed set in MD 32060: POS_AX_VELO is applied.
- The programmed instruction is effective on a non-modal basis. The geometry and synchronous axes are separated with the instructions from the path axis grouping and traversed at an axis-specific velocity.

**With SW 5 and higher**, type 1 positioning axes have the following additional behavior:

- With continuous-path mode (G601, G602, G603) positioning axes/spindles traverse to the positioning end if motion condition FINEA, COARSA or IPOENDE is fulfilled.

**Positioning axis type 2**

**Block change at programmed end point of all path axes**

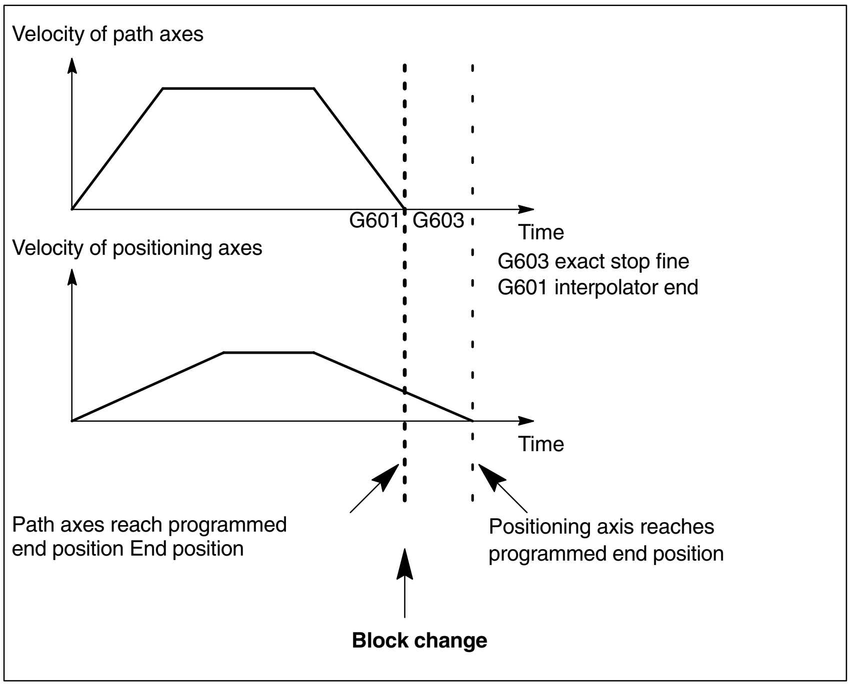



Fig. 2-2 Block change with positioning axis type 2, example of sequence

**Properties of type 2 positioning axis**

With SW 5 and lower, type 2 positioning axes have the following behavior:

- The block change occurs (NC block finished) when the path axes have reached their programmed end positions with regard to G601, G602 and G603.
- The positioning axes can traverse across several block boundaries to their programmed end positions.
- Since there is no time relationship between “NC block finished” and the point at which type 2 positioning axes reach their programmed end positions, the WAITP coordination command is provided for the synchronization of the positioning axes (see Section 2.6).
- If a positioning axis is reprogrammed before it has reached the previous position and before the motion end condition is fulfilled, the “axis cannot be repositioned” alarm is issued.
- Programming with
  POSA[name] = end point FA[name] = feed
  or abbreviated with
  POSA[name] = end point
  in which case the feed set in MD 32060: POS_AX_VELO is applied.
- The programmed instruction is effective on a non-modal basis. The geometry and synchronous axes are separated with the instructions from the path axis grouping and traversed at an axis-specific velocity.

### 2.3.1 Settable block change time (SW 6.2 and higher)

**Positioning axis type 3**

**Settable block change time for single axis interpolation**

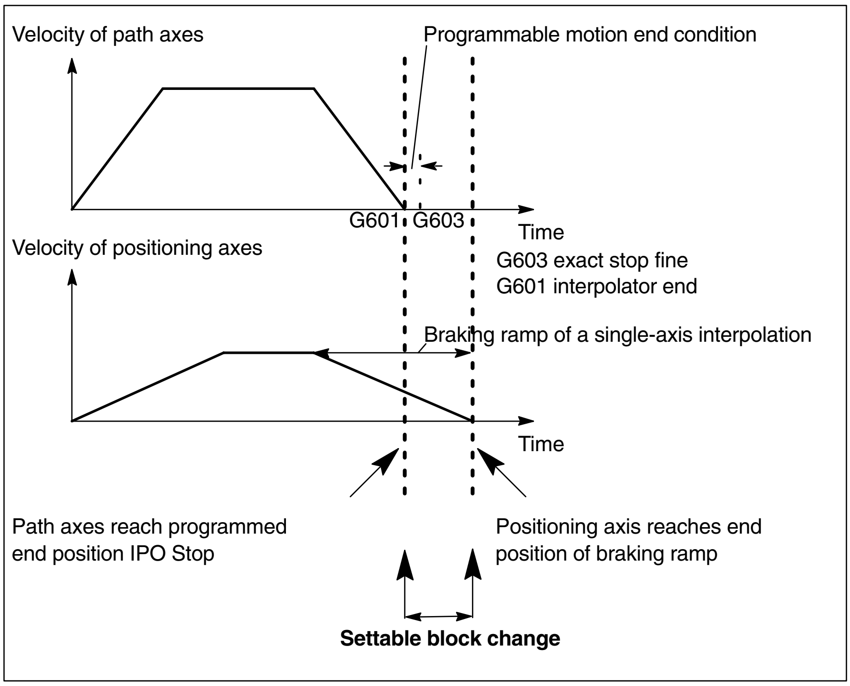



Fig. 2-3 Settable block change for type 3 positioning axis, example

**Properties of type 3 positioning axis**

With type 3 positioning axes the motion end condition can always be programmed with FINEA, COARSEA or IPOENDA. The block change condition can be set within the braking ramp of the single-axis interpolation.

- The block change occurs (NC block finished) when all the path axes/spindles have fulfilled the programmed motion end conditions. The positioning axis/spindle must also have met the programmed block change condition within the braking ramp of the single-axis interpolation. The block change can only be executed if both conditions are fulfilled.
- The position end of the positioning axes can be defined in the programmable motion end condition by means of one of the following NC commands
  FINEA[axis]: “Exact stop fine” motion end or
  COARSEA[axis]: “Exact stop coarse” motion end or
  IPOENDA[axis]: End of motion when “IPO stop” is reached.
- The following syntax applies for the position end of positioning spindles:
  FINEA[Sn]: “Exact stop fine” motion end or
  COARSEA[Sn] “Exact stop coarse” motion end or
  IPOENDA[Sn]: End of motion when “IPO stop” is reached

  Sn: Spindle number, 0... max. spindle number or
  axis: Axis identifier, X, Y, Z.

**Command axes**

Command axes respond immediately to any reprogramming; therefore, programming should not be changed. Each individual axis has its own axis interpolator and its own feedrate. An axis cannot be moved by the parts program and synchronized actions simultaneously.

The FINEA, COARSEA, IPOENDA motion end conditions are applicable for the transition to a neutral axis. The axis can then be used via the NC program, for example, as a PLC axis.

**Reciprocating axes**

Reciprocating axes always brake at their reversal position and then move in the opposite direction. Therefore, reciprocating axes do not require an expansion.

---

**Note**

The behavior of PLC axes at block change is described in Section 2.5 Control by PLC.
For further information about block change with programmable motion end conditions FINEA, COARESA and IPOENDA, please refer to:

**References:** /FB/, S1, “Spindles”, Spindle modes
/FB/, V1, “Feeds”, Programmable single-axis dynamic response

---

**IPOBRKA**

When the NC command IPOBRKA is issued, the motion end condition is activated in the braking ramp either

- in the parts program: Block change in the parts program or
- in synchronized action: Technology cycles

**With SW 6.2 and higher, type 2** positioning axis behavior differs from that in previous software versions as follows:

If POSA if programmed, then POSA again with IPOBRKA (block change in the braking ramp), an alarm is not issued. The new POSA motion waits for the preceding POSA motion to have reached the block change point in the braking ramp.

**Condition for block change**

If the motion end conditions for all axes/spindles to be operated in the block are fulfilled in addition to the block change condition, then block change takes place. This applies both to parts program blocks and for technology cycle blocks.

**Braking ramp block change condition in SD 43600**

The new setting data “Block change condition braking ramp”
SD 43600: IPOBRAKE_BLOCK_EXCHANGE
allows the block change time to be set for single-axis interpolation from the start time of the braking ramp at 100% right up to IPO_Stop at 0%.

With 100% the block change condition of the positioning axis is already fulfilled at the start point of the braking ramp.
With 0% the block change condition is identical to IPOENDA.

**Advantages of the percent setting**

Setting SD 43600 in percent offers the following advantages:

- The block change condition is not dependent on a position and is therefore dependent on the override set.
- Maximum override will result in the greatest smoothing deviation.
- A smaller override will reduce the deviation.

The braking ramp block change condition can be queried like the previous axial motion end conditions with $AA_MOTEND[axis] = 4

**Braking ramp with tolerance window**

**Additional block change condition in the braking ramp with SW 6.4 and higher**

From SW 6.4, there will be an additional tolerance window to be selected as well as the already existing block change condition in the braking ramp. Release will only occur when the axis

- as before has reached the specified % value of its braking ramp **and**
- from SW 6.4, its current actual position or setpoint position is no further away than an adjustable tolerance from the end position of the axis in the block.

The latest the axis is available is when the interpolator reaches the end position. The tolerance window is modal.

The following applies to the braking ramp **with tolerance window** block change condition:

The setpoint position is notified with $AA_MOTEND[axis] = 5
The actual position is notified with $AA_MOTEND[axis] = 6

**Supplementary conditions**

**Block change and alteration of axis status**

A premature block change is not possible:

- During oscillation with partial infeed
  the block-specific oscillation motion must remain active until the axis with partial infeed has reached its end position.
- With handwheel traversal
  the last set end of motion criterion is operative.

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.

At axis state change, e.g. from POS to SPOS, the last programmed motion end condition FINEA, COARSEA or IPOENDA is active.
This also applies:

- if a positioning axis is changed to a path axis or
- if the program is specifically waiting for the end of the positioning motion (WAITP, M30, technology cycle end, preprocessing stop)
- if a velocity override is activated or deactivated.

**Activation and deactivation**

**Motion end condition IPOBRKA and precise time of activation.**

For parts program execution:

- The braking ramp motion end condition can be activated via the NC command IPOBRKA.
- The precise time of activation is defined in setting data SD 43600: IPOBRAKE_BLOCK_EXCHANGE.
- IPOBRKA is deactivated when an axis motion end condition FINEA, COARSEA or IPOENDA is next programmed for the axis.
- **With SW 6.4 and higher**
  The NC command ADISPOSA can be used to program the size of the tolerance window for the braking ramp motion end condition and this is defined by means of setting data SD 43610: ADISPOSA_VALUE[axis].

For technology cycles:

- The braking ramp motion end condition can be activated via the synchronous action IPOBRKA.
- IPOBRKA is deactivated when an axis motion end condition FINEA, COARSEA or IPOENDA is next programmed for the axis in a synchronous action.
- **With SW 6.4 and higher**
  The size of the tolerance window for the braking ramp motion end condition can be programmed via the synchronous action ADISPOSA.

---

**Note**

For more information on the programming of positioning axes, see:
**References:** /PG/, Chapter 7, “Feedrate regulation and spindle motion”
/PGA/, Chapter 5, “Special motion commands”

---

**Examples**

For block change condition “Braking ramp” in the **parts program**:

```
                     ; Default effective
N10 POS[X] = 100     ; Block change occurs if the X axis is at position 100
                     ; and has reached exact stop fine.

N20 IPOBRKA(X,100)   ; Braking ramp block change condition
N30 POS[X] = 200     ; Block change occurs as soon as the X axis starts braking
N40 POS[X] = 250     ; X axis does not brake at position 200 but continues traversing
                     ; to position 250, as soon as the X axis starts braking
                     ; block change occurs.

N50 POS[X] = 0       ; The X axis brakes and traverses back to position 0
                     ; block change occurs at position 0 exact stop fine
N60 X10 F100
N70 M30
```

At direction reversal (N50), the axis first brakes to reach the target position, before it can be traversed in the opposite direction.

Alternatively, IPOBRKA(X) can also be written in block N20 if the value 100 has already been entered in SD 33500: IPOBRAKE_BLOCK_EXCHANGE[AX1].

For block change condition “Braking ramp” in **synchronized action**:

In the technology cycle:

```
FINEA                   ; End of motion criterion 'exact stop fine'
POS[X] = 100            ; Technology cycle block changes if the
                        ; X axis has reached position 100 and
                        ; exact stop fine.

IPOBRKA(X,100)          ; Block change criterion 'activate braking ramp'
POS[X] = 200            ; Technology cycle block changes as soon
                        ; as the X axis starts to brake.

POS[X] = 250            ; X axis does not brake at position 200 but continues traversing
N40 POS[X] = 250        ; to position 250, as soon as the X axis starts braking
                        ; block change occurs in the technology cycle.

POS[X] = 0              ; X axis brakes and moves back to position 0
M17                     ; block change occurs at position 0 and exact stop fine.
```

**With tolerance window SW 6.4 and higher**

For block change condition “Braking ramp” in the **parts program**:

```
                        ; Default effective
N10 POS[X] = 100        ; Block change occurs if the X axis is at position 100
                        ; and has reached exact stop fine.

N20 IPOBRKA(X,100)      ; Braking ramp block change condition
N21 ADISPOSA(X,1,0.5)   , define tolerance window setpoint position for X axis with 0.5
N30 POS[X] = 200        ; Block change occurs as soon as the X axis starts braking
                        ; and the setpoint position is ≥ 199.5.
N40 POS[X] = 250        ; X axis does not brake at position 200 but continues traversing
                        ; to position 250, as soon as the X axis starts braking
                        ; and the position is ≥ 249.5, the block change occurs.

N50 POS[X] = 0          ; The X axis brakes and traverses back to position 0
                        ; block change occurs at position 0 exact stop fine
N60 X10 F100
N70 M30
```

At direction reversal (N50), the axis first brakes to reach the target position, before it can be traversed in the opposite direction.

For block change condition “Braking ramp” in **synchronized action**:

In the technology cycle:

```
FINEA                   ; End of motion criterion 'exact stop fine'
POS[X] = 100            ; Technology cycle block changes if the
                        ; X axis has reached position 100 and
                        ; exact stop fine.

ADISPOSA(X,2,0.3)       ; Define tolerance window actual position for X axis with 0.3
IPOBRKA(X,100)          ; Block change criterion 'activate braking ramp'
POS[X] = 200            ; Technology cycle block change occurs
                        ; as soon as the X axis starts to brake and
                        ; the actual position of the X axis is ≥ 199.7.

POS[X] = 250            ; X axis does not brake at position 200 but continues traversing
N40 POS[X] = 250        ; to position 250, as soon as the X axis starts braking
                        ; and the position is ≥ 249.7, the block change occurs in the
                        ; technology cycle.

POS[X] = 0              ; X axis brakes and moves back to position 0
M17                     ; block change occurs at position 0 and exact stop fine.
```

### 2.3.2 End of motion criterion with block search (SW 6.1 and higher)

**Last block serves as container**

The last movement end 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 motion end condition for three axes:

```
N01 G01 POS[X]=20 POS[Y]=30
      IPOENDA[X]            ; Last programmed motion end condition IPOENDA
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 a digital output
N05 POS[X]=100              ;
N06 IPOBRKA(X, 100)         ; Second action block for the X axis IPOBRKA container
...                         ; for all programmed motion end conditions
TARGET:                     ; Block search target
```

The digital output:
$A_OUT[1]=1 is output in the first action block.

The motion end conditions are set:
for X axis IPOBRKA / $SA_IPOBRAKE_BLOCK_EXCHANGE[AX1]=100
for Y axis IPOBRKA / $SA_IPOBRAKE_BLOCK_EXCHANGE[AX2]=50
for Z axis FINEA in the second action block. The motion end condition IPOENDA last programmed is also stored for the X axis.

## 2.4 Velocity

The axis-specific velocity limits and acceleration limits are valid for positioning axes.

Positioning axes can be linear axes and rotary axes.

Positioning axes can also be indexing axes, see:
**References:** /FB/, T1, “Indexing axes”

**Feedrate override**

The path and positioning axes have separate feedrate overrides. Each positioning axis can be adjusted by its own axis-specific feed 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.

**Feed**

The positioning axes traverse at the axis-specific feedrate programmed for them. As illustrated in Section 2.2, the 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 MD 32060: POS_AX_VELO (initial setting for positioning axis velocity).

The programmed axis-specific feedrate is active until the end of the program.

**Revolutional feedrate**

In JOG mode, the response of the axis/spindle is also dependent on the setting in SD 41100: JOG_REV_IS_ACTIVE (revolutional feedrate for JOG active).

- If this setting data is active, an axis/spindle is always traversed at revolutional feedrate MD 32050: JOG_REV_VELO (revolutional feedrate for JOG) or MD 32040: JOG_REV_VELO_RAPID (revolutional feedrate for JOG with rapid traverse overlay) depending on the master spindle.
- If the setting data is not active, then the axis/spindle responds as a function of the setting in SD 43300: ASSIG_FEED_PER_REV_SOURCE (revolutional feedrate for positioning axes/spindles).
- If the setting data is not active, then a geometry axis influenced by a frame with rotation responds as a function of on channel-specific setting data SD 42600: JOG_FEED_PER_REV_SOURCE. (In the operating mode JOG, revolutional feedrate for geometry axes on which a frame with rotation is effective).

## 2.5 Control by PLC

**Channel-specific signals**

All channel-specific signals act to the same extent on path and positioning axes.

The following signals are the only exceptions (see Chapter 5):

- IS “Feedrate override” (DB21, ... DBB4)
- IS “Delete distancetogo” (DB21, ... DBX6.2)

**Axis-specific signals**

The following additional signals are available for positioning axes (see Chapter 5):

- IS “Positioning axis” (DB31, ... DBX76.5)
- F function (feed) for FA positioning axes
- Feedrate override, axis-specific
- IS “Delete distancetogo” (DB31, ... DBX2.2) axis-specific

**Parameters for FC15**

When concurrent positioning axes are activated from the PLC, FC15 is called and supplied with the following parameter data:

- Axis name/axis number
- End position
- Feedrate (with feedrate setting = 0, the feed set in MD 32060: POS_AX_VELO) is applied
- Absolute coordinates (G90), incremental coordinates (G91), absolute coordinates 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

**PLC axes**

PLC axes are traversed in the basic program from the PLC via special function blocks and can move asynchronous with all other axes. The travel motions are executed separately from the path and synchronized actions.

## 2.6 Programming

**Note**

The following documentation should be consulted for instructions on programming positioning axes:
**References:** /PA/, “Programming Guide”

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: Type 1 or type 2 positioning axis and type 3 with SW 6.2 and higher
- End point dimensions
- Absolute or incremental dimension for the end position coordinates
- Feedrate for linear axes in [mm/min], for rotary axes in [degrees/min]

**Syntax**

Positioning axis type 1:

POS[Q1]=200 FA[Q1]=1000; axis Q1 with feedrate 1,000mm/min to position 200

Positioning axis type 2:

POSA[Q2]=300 FA[Q2]=1500; axis Q2 with feedrate 1500 mm/min to position 300

Within a parts 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.

**Absolute/ incremental dimensions**

The end position coordinates are programmed as absolute dimensions (G90) or incremental dimensions (G91):

**Absolute dimensions**

G90 POS[Q1]=200
G91 POS[Q1]=AC(200)

**Incremental dimension**

G91 POS[Q1]=200
G90 POS[Q1]=IC(200)

**Reprogram type 2 positioning axes**

With type 2 positioning axes (motion across block limits), you need to be able to detect in the parts 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 if programmed, then POSA again with IPOBRKA (block change in the braking ramp SW 6.2 and higher), an alarm is not issued. For more information, please refer to Subsection 2.3.1, NC command IPOBKA.

**WAITP coordination**

The WAITP coordination command is used to query in the parts program whether the end position has been reached.

WAITP is programmed in a separate block.

An explicit reference must be made to any axis for which the program is to wait.

Example program:

```
N10 G01 G90 X200 F1000 POSA[Q1]=200 FA[Q1]=500
N15 X400
N20 WAITP(Q1);      Execution of the program stops automatically
                    until Q1 in position
N25 X600 POS[Q1]=300; Q1 is positioning axis type 1 (feed 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.

## 2.6.1 Programming from external

Traversing at revolutional feedrate from an external source can be selected via axial data SD 43300: ASSIGN_FEED_PER_REV_SOURCE, (revolutional feedrate for axes) and channel-specific setting data SD 42600: JOG_FEED_PER_REV_SOURCE in JOG mode. The following settings can be made via the setting data:

- >0: The machine axis number of the rotary axis/spindle from which the revolutional feedrate shall be derived
- –1: The revolutional feedrate is derived from the master spindle of the channel in which the axis/spindle is active in each case
- 0: The function is deselected

# 2.7 Response with special functions

## 2.7.1 Dry run feedrate (DRY RUN)

The dry run feedrate is also effective for positioning axes unless the programmed feedrate is larger than the dry run feedrate. (SW version 5 or earlier).

From SW version 6, the effectiveness of the dry run feedrate set in SD 42100: DRY_RUN_FEED can be controlled with SD 42101: DRY_RUN_FEED_MODE. See

**References:** /FB1/, V1, Feedrates

## 2.7.2 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.

■

# Supplementary Conditions 3

There are no supplementary conditions stipulated for this Description of Functions.

■

# Data Descriptions (MD, SD) 4

## 4.1 General machine data

| | | | | | |
|---|---|---|---|---|---|
| **10008**<br>MD number | **MAXNUM_PLC_CNTRL_AXES**<br>Max. no. PLC-controlled axes | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit:<br>12 (NCU573),<br>otherwise 4 | |
| Changes effective after POWER ON | | | Protection level: 2/7 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 6.3 | |
| Meaning: | Maximum number of axes that can be controlled by the PLC. | | | | |

# 4.2 Channelspecific machine data

| | | | | | |
|---|---|---|---|---|---|
| **20730**<br>MD number | **G0_LINEAR_MODE**<br>Interpolation behavior with G0 | | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: 2/7 | | Unit: – |
| Data type: BOOLEAN | | | | Applies from SW: SW 6.1 | |
| Meaning: | This machine data is used to define the interpolation behavior at rapid traverse velocity (G0):<br>0: Non-linear interpolation:<br>Each path axis interpolates as a single axis (positioning axis) independently of the other axes at the rapid traverse velocity (G0)<br>defined for the axis (MD 32000: MAX_AX_VELO).<br>1: Linear interpolation:<br>The path axes are interpolated together.<br>The non-linear interpolation can be activated via the NC parts program G0LINOF, and deactivated via G0LINOF. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **20732**<br>MD number | **EXTERN_G0_LINEAR_MODE**<br>Interpolation behavior with G00 | | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: 2/7 | | Unit: – |
| Data type: BOOLEAN | | | | Applies from SW: SW 6.1 | |
| Meaning: | This machine data is used to define the interpolation behavior with G00:<br>0: Axes are traversed as positioning axes<br>1: Axes interpolate with one another | | | | |
| MD irrelevant for ... ... | **In SW 6 and higher**, the machine data<br>MD 10892: EXTERN_G00_MODE used for ISO mode up to now has been replaced with the new machine data<br>MD 20732: EXTERN_G0_LINEAR_MODE. | | | | |

## 4.3 Axis/spindle-specific machine data

| | | | |
|---|---|---|---|
| **30450**<br>MD number | **IS_CONCURRENT_POS_AX**<br>Default on RESET: Neutral axis or channel axis | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after RESET | Protection level: 2/7 | Unit: none | |
| Data type: BOOLEAN | Applies from SW: 1.1 | | |
| Meaning: | The axis is a concurrent positioning axis.<br>SW 4.3 and higher (not FM-NC):<br>If FALSE: At RESET a neutral axis becomes a channel axis again.<br>If TRUE: At RESET a neutral axis remains in the neutral axis state, and a channel axis becomes a neutral axis. | | |

| | | | |
|---|---|---|---|
| **32060**<br>MD number | **POS_AX_VELO**<br>Initial setting for positioning axis velocity | | |
| Default setting: 100000 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after RESET | Protection level: 2/7 | Unit: mm/min<br>rev/min | |
| Data type: DOUBLE | Applies from SW: 1.1 | | |
| Meaning: | Where a positioning axis is programmed in the part program without specifying the axis-specific feedrate, the feedrate entered in MD: POS_AX_VELO is automatically used. The feedrate from MD: POS_AX_VELO applies until an axis-specific feedrate is programmed in the parts program for this positioning axis. | | |
| MD irrelevant for ... ... | POS_AX_VELO is irrelevant as a positioning axis for all other axis types. | | |
| Special cases, errors, ... ... | If a zero velocity setting is entered in POS_AX_VELO, the positioning axis does not traverse if it is programmed without feed. If a velocity setting is entered in POS_AX_VELO that is higher than the maximum velocity of the axis (MD 32000: MAX_AX_VELO), the velocity is automatically restricted to the maximum rate. | | |

| | | | |
|---|---|---|---|
| **37510**<br>MD number | **AX_ESR_DELAY_TIME1**<br>Delay time for ESR single axis | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: time | |
| Changes effective after NEW_CONF | Protection level: 2/7 | Unit: – | |
| Data type: DOUBLE | Applies from SW: 6.4 | | |
| Meaning: | When an alarm occurs, it is possible to use the existing machine data to delay the instant of braking, so that in the case of gear wheel cutting, for example, it is possible to retract from the tooth gap. | | |

| | | | |
|---|---|---|---|
| **37511**<br>MD number | **AX_ESR_DELAY_TIME2**<br>ESR time for interpolatory braking of the single axis | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: time | |
| Changes effective after NEW_CONF | Protection level: 2/7 | Unit: – | |
| Data type: DOUBLE | Applies from SW: 6.4 | | |
| Meaning: | Once time MD 37510: AX_ESR_DELAY_TIME1 has expired, the time specified here (MD 37511: AX_ESR_DELAY2) for interpolatory braking is still available.<br>Once time MD 37511: AX_ESR_DELAY2 has expired, rapid deceleration with subsequent tracking is initiated. | | |

## 4.4 Axis-specific setting data

| | | | | | |
|---|---|---|---|---|---|
| **43600**<br>SD number | **IPOBRAKE_BLOCK_EXCHANGE**<br>Braking ramp block change condition | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 100 | |
| Changes effective after: immediately | | | Protection level: 7/7 | | Unit: % |
| Data type: DOUBLE | | | | Applies from SW: 6.1 | |
| Meaning: | Specifies the time of activation for the braking ramp block change condition with single-axis interpolation:<br>100% signifies that the block change condition for activating the braking ramp is fulfilled.<br>0% signifies that the block change condition is identical to IPOENDA. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **43610**<br>SD number | **ADISPOSA_VALUE**<br>Braking ramp tolerance window | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after: immediately | | | Protection level: 7/7 | | Unit: – |
| Data type: DOUBLE | | | | Applies from SW: 6.4 | |
| Meaning: | With single axis interpolation, the value defines the size of the tolerance window that has to be reached by the axis in order to enable a block change when the braking ramp with tolerance window block change condition is valid and the corresponding % value of the braking ramp, SD 43600: IPOBRAKE_BLOCK_EXCHANGE has been reached.<br><br>**Note:**<br>The machine data MD 10710: PROG_SD_RESET_SAVE_TAB can be set such that the value written from the parts program during reset is accepted by the active file system (i.e. the value is retained beyond the reset). | | | | |

■

# Signal Descriptions 5

The following signals or commands on the NCK/MMC/PLC interface are only of significance for the positioning axis:

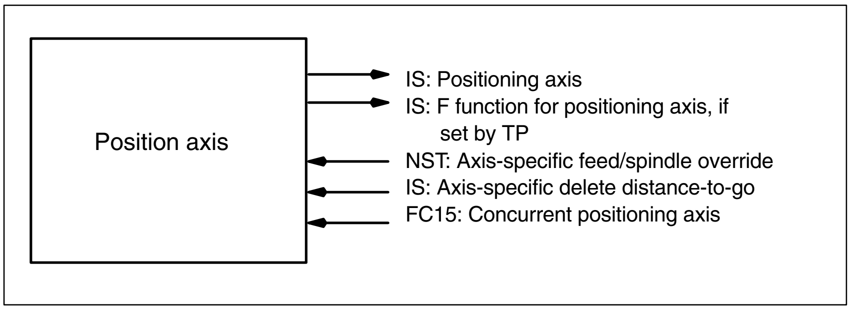



Fig. 5-1 Signal modification by the PLC

## 5.1 Axis/spindle-specific signals

| | |
|---|---|
| **DB31, ...**<br>**DBB0**<br>Data Block | **Feedrate override/spindle speed override, axis-specific**<br><br>Signal(s) from axis/spindle (NCK ——> PLC) |
| Edge evaluation: no | Signal(s) updated: Cyclic — Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | Positioning axes have their own axis-specific feed override value. This feedrate override is evaluated in the same way as the channel-specific feedrate override. |
| Signal irrelevant for ...... | IS “Positioning axis” (DB31, ... DBX74.5) = ZERO |
| References | For evaluation see IS “Feedrate override” (DB21, ... DBB4) channel-specific |

| | |
|---|---|
| **DB31, ...**<br>**DBX2.2**<br>Data Block | **Delete distance-to-go, axis-specific**<br><br>Signal(s) from axis/spindle (NCK ——> PLC) |
| Edge evaluation: yes | Signal(s) updated: Cyclic — Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 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. |
| Related to .... | IS “Delete distancetogo” (DB21, ... DBX6.2) channel-specific for path axes |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX28.1**<br>Data Block | **AxReset**<br>Reset axis/spindle<br>Signal(s) from axis/spindle (PLC ––> NCK) | |
| Edge evaluation: yes | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.4 |
| Signal state 1 or signal transition 0 ––> 1 | An axis/spindle controlled from the PLC is to be reset again.<br>The NCK switches the active axis/spindle to the stop state with IS “AXSTOP active” (DB31 ... DBX63.2). | |
| Special cases, errors, ...... | Supplementary condition:<br>The axis/spindle must have actually accepted the PLC and be controlled by the PLC. | |
| Related to .... | IS “AXRSTOP” (DB31, ... DBX63.2) Stop axis/spindle | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX28.2**<br>Data Block | **AxResume**<br>Resume axis/spindle motion<br>Signal(s) from axis/spindle (PLC ––> NCK) | |
| Edge evaluation: yes | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.4 |
| Signal state 1 or signal transition 0 ––> 1 | An interpolating axis/spindle from the main run continues independently of the NC program and is controlled by the PLC.<br>The NCK checks whether an axis/spindle axial alarm with the cancel criterion CANCEL-CLEAR or NCSTARTCLEAR is active and deletes it.<br>The axis-specific RESUME (DB31 ... DBX28.2) instruction can be aborted by the NCK with IS “AXSTOP active” (DB31 ... DBX28.2). | |
| Special cases, errors, ...... | Supplementary condition:<br>The axis/spindle must be operating under PLC control.<br>The IS “AXRESUME” (DB31 ... DBX28.2) is ignored in the following error states detected by the NCK:<br>1. When the PLC prompts axes/spindles to continue traversing and this axis/spindle has not been accepted by the PLC.<br>2. The axis/spindle is not in the stopped state.<br>3. The axis/spindle cannot resume traversal because an alarm is active. | |
| Related to .... | IS “AXRESET” (DB31, ... DBX28.1) Reset axis/spindle | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX61.1**<br>Data Block | **AxAlarm**<br>Axial stop alarm for this axis<br>Signal(s) from axis/spindle (NCK ––> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.4 |
| Signal state 1 or signal transition 0 ––> 1 | NCK decelerates the axis/spindle down a ramp and confirms the braking operation via the OPI.<br>At the same time, an alarm is signaled to the PLC with<br>IS “Axial alarm” (DB31 ... DBX61.1) == 1 and the status system variable $AA_SNGLAX_STAT == 5 is set. | |
| Related to .... | System variable $AA_SNGLAX_STAT == 5 | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX63.0**<br>Data Block | **AxReset done**<br>Axial reset has been performed<br>Signal(s) from axis/spindle (NCK ––> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.4 |
| Signal state 1 or signal transition 0 ––> 1 | After the axial machine data active on Reset have been activated,<br>IS “AXRESET DONE” (DB31 ... DBX63.0) == 1,<br>IS “Axstop active” (DB31 ... DBX63.2) == 0 and<br>system variable $AA_SNGLAX_STAT == 1 “single axis in reset active” are signaled. | |
| Related to .... | IS “ Axstop active” (DB31 ... DBX63.2) == 0<br>System variable $AA_SNGLAX_STAT == 1 | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX63.1**<br>Data Block | **PLC controls axis**<br>Axis control passed to PLC<br>Signal(s) from axis/spindle (NCK ––> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.4 |
| Signal state 1 or signal transition 0 ––> 1 | IS “PLC control axis” (DB31 ... DBX63.1) == 1 is passed to the PLC to signal the status of the axis/spindle and confirm that control has been accepted by the NCK.<br>The current axis status can be scanned with $AA_SNGLAX_STAT. | |
| Related to .... | IS “PLC controls axis” (DB31 ... DBX28.7) Axis is controlled by the PLC.<br>System variable $AA_SNGLAX_STAT Current axis status | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX63.2**<br>Data Block | **AxStop active**<br>Acknowledgement of stopped status<br>Signal(s) from axis/spindle (NCK ––> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.4 |
| Signal state 1 or signal transition 0 ––> 1 | NCK switches the axis/spindle to the stop state by setting IS “AXSTOP ACTIVE” (DB31 ... DBX63.2) == 0. All axis motions controlled by the main run are stopped.<br>Axis status interrupted with system variable $AA_SNGLAX_STAT == 3. | |
| Related to .... | IS “AXSTOP, stop” (DB31 ... DBX28.6) Stop axis/spindle<br>System variable $AA_SNGLAX_STAT == 3 | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX76.5**<br>Data Block | **Positioning axis**<br>NCK treats the axis as a positioning axis.<br>Signal(s) from axis/spindle (NCK ––> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The NCK treats the axis as a positioning axis. It therefore has:<br>a dedicated axis interpolator (linear interpolator)<br>a dedicated feedrate (F value)<br>a dedicated feed override<br>exact stop (G09) at the programmed end position End position | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBB78–81**<br>Data Block | **F function (feedrate) for positioning axis**<br>Assign feedrate through the programmed axis name of a positioning axis<br>Signal(s) from axis/spindle (NCK ––> PLC) | |
| Edge evaluation: no | Signal(s) updated: when change | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ––> 1 | The programmed axial feed is assigned to a positioning axis by means of the programmed axis name and output to the PLC for this axis. There is no output of the value preset via FC15. See below. | |
| Signal irrelevant for ...... | IS: Positioning axis = ZERO | |
| Special cases, errors, ...... | If the positioning axis is traversed at the feedrate set in MD 32060: POS_AX_VELO (initial setting for positioning axis velocity), the NC does not output an F function (feed) to the PLC. | |
| Related to .... | IS “Positioning axis” (DB31, ... DBX74.5)<br>MD 22240: AUXFU_F_SYNC_TYPE Output time of the F functions | |

## 5.2 Function call

**FC15**

PLC function call FC15 can be used to start concurrent positioning axes from the PLC. The following parameters are passed to the function call:

- Axis name/axis number
- End position
- Feedrate (with feedrate setting = 0, the feed set in MD 32060: POS_AX_VELO is applied).
  The F value of FC15 is **not** transferred to the axis-specific IS “F function (feedrate) for positioning axis” DB31, ...DBB78-81.
- 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.

**References:** /FB/, P3, “Basic PLC Program”

■

# Example 6

## 6.1 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).

**Programming example**

```
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
```

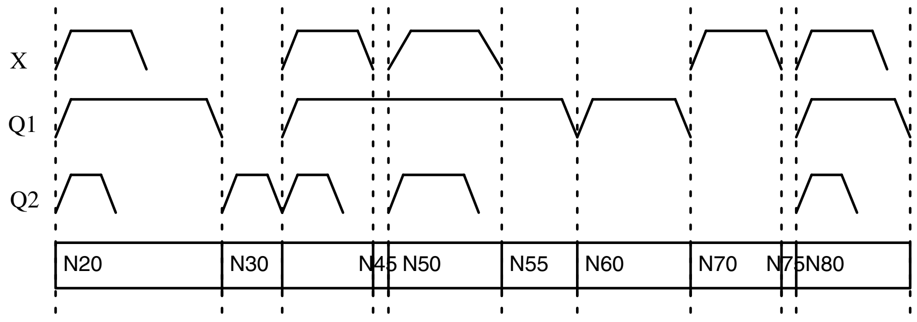



Fig. 6-1 Timing of path axes and positioning axes

### 6.1.1 Path axes traverse in G0 with no interpolating (SW 6.1 and higher)

**Example in G0 for positioning axes**

Path axes traverse as positioning axes with no interpolation in rapid traverse mode (G0):

| | |
|---|---|
| | ; Activation of non-linear interpolation |
| | ; MD 20730: GO_LINEAR_MODE == FALSE is set |
| G0 X0 Y10 | ; traversal **without interpolation** |
| G0 G43 X20 Y20 | ; traversal in path mode (with interpolation) |
| G0 G64 X30 Y30 | ; traversal in path mode (with interpolation) |
| G0 G95 X100 Z100 m3 s100 | ; traversal **without interpolation** |
| | ; no revolutional feedrate active |

## 6.2 Examples of autonomous single axis operations (SW 6.3 and higher)

### 6.2.1 PLC actions as NCK reaction (SW 6.4 and higher)

**Example of a machine axis**

PLC actions are shown below as NCK reactions

| PLC actions | NCK reaction |
|---|---|
| **Start machine axis 1, residing in 1st channel, as PLC axis via FC 18** | |
| IS “NC Stop plus spindle” DB21, ... DBX7.4 | Machine axis 1 is stopped |
| Trigger IS “NC Start” (DB21, ... DBX7.1) | Machine axis 1 continues traversal |
| PLC wants to control machine axis 1, IS “PLC controls axis” DB31, ... DBX28.7==1 | Control of machine axis 1 is given to the PLC. IS “PLC controls axis” DB31, ... DBX63.1==1 |
| Trigger IS “NC Stop plus spindle” DB21, ... DBX7.4 | Machine axis 1 is **not** stopped. |
| Trigger axial axis stop IS “AxStop, stop” DB31, ... DBX28.6 | Machine axis 1 is stopped IS“AxStop active” DB31, ... DBX63.2==1 |
| Trigger axial resume IS “AxResume” DB31, ... DBX28.2 | Machine axis 1 continues traversal IS “AxStop active” DB31, ... DBX63.2==0 |
| Trigger NC reset Trigger IS “Reset” DB21, ... DBX7.7 | No effect on machine axis 1 |
| Trigger axial reset IS “AxReset” DB31, ... DBX28.1 | Machine axis 1 is stopped IS “AxStop active” DB31, ... DBX63.2==1 is reset to 0, its axial machine data are read in, IS “AxReset done” DB31, ... DBX63.0 is set to 1 and IS “AxStop active” DB31, ... DBX63.2 is reset to 0. |

| **Start machine axis 1 as PLC axis via FC 18** | IS “AxReset done”<br>DB31, ... DBX63.0==0 |
|---|---|
| Cancel servo enable for machine axis 1<br>IS “servo enable” DB31, ... DBX2.1==0 | Axial alarm 21612 “Axis %1 measuring system change” signaled. |
| Trigger axial resume<br>IS “AxResume” DB31, ... DBX28.2 | Axial alarm 21612 “Axis %1 measuring system change” is canceled and motion command IS “Motion command plus” DB21, ... DBX40.7 is output. But machine axis 1 does not start to traverse due to a missing servo enable. |
| Set servo enable for machine axis 1 IS “servo enable” DB31, ... DBX2.1==1 | Machine axis 1 traverses to the programmed end point. |
| Trigger axial reset<br>IS “AxReset” DB31, ... DBX28.1 | Machine axis 1 is reset internally, its axial machine data are read in and the end of axial reset IS “AxReset done” DB31, ... DBX63.0==0 is signaled. |
| PLC releases control of machine axis 1 to the NCK. IS “PLC controls axis” DB31, ... DBX28.7==0 | NCK accepts control of machine axis 1 IS “PLC controls axis” DB31, ... DBX63.1==0 and resets the end signal of axial reset IS “AxReset done” DB31, ... DBX63.0 from 1 to 0. |

■

# Notes

# Data Fields, Lists 7

## 7.1 Interface signals

| DB number | Bit, byte | Name | Refer-ence |
|---|---|---|---|
| **Channel-specific** | | | |
| 21–30 | 7.1 | NC Start | K1 |
| 21–30 | 7.4 | NC stop axes plus spindle | K1 |
| 21–30 | 7.7 | Reset | K1 |
| 21–30 | 40.6 | Motion command minus | H1 |
| 21–30 | 40.7 | Motion command plus | H1 |
| **Axis/spindle-specific** | | | |
| 31–61 | 0 | Feedrate override, axis-specific | V1 |
| 31–61 | 2.1 | Controller enable | A2 |
| 31–61 | 2.2 | Delete distance-to-go spindle reset for specific axes | A2, S1 |
| 31–61 | 28.1 | AxReset | |
| 31–61 | 28.2 | AxResume | |
| 31–61 | 28.6 | AxStop, stop | P5 |
| 31–61 | 28.7 | PLC controls axis | P5 |
| 31–61 | 60.6 | Exact stop coarse | B1 |
| 31–61 | 60.7 | Exact stop fine | B1 |
| 31–61 | 61.1 | AxAlarm | |
| 31–61 | 61.2 | Axis ready (AX_IS_READY) | B3 |
| 31–61 | 62.7 | Axis container rotation active | B3 |
| 31–61 | 63.0 | AxReset done | |
| 31–61 | 63.1 | PLC controls axis | |
| 31–61 | 63.2 | AxStop active | |
| 31–61 | 64.6 | Motion command minus | H1 |
| 31–61 | 64.7 | Motion command plus | H1 |
| 31–61 | 76.5 | Position axis | |
| 31–61 | 78–81 | F function (feedrate) for positioning axis | V1 |
| 31–61 | 98.7 | Emergency retraction active | |

## 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ...** | | | |
| 10008 | MAXNUM_PLC_CNTRL_AXES | Max. no. PLC-controlled axes (SW 6.3 and higher) | |
| **Channelspecific ($MC_ ... )** | | | |
| 20730 | G0_LINEAR_MODE | Interpolation behavior with G0 (SW 6.1 and higher) | |
| 20732 | EXTERN_G0_LINEAR_MODE | Interpolation behavior with G00 (SW 6.1 and higher) | |
| 22240 | AUXFU_F_SYNC_TYPE | Output timing of F functions | H2 |
| **Axisspecific ($MA_ ... )** | | | |
| 30450 | IS_CONCURRENT_POS_AX | Concurrent positioning axis | |
| 32060 | POS_AX_VELO | Feedrate for positioning axis | |
| 37510 | AX_ESR_DELAY_TIME1 | Delay time for ESR single axis (SW 6.4 and higher) | |
| 37511 | AX_ESR_DELAY_TIME2 | ESR time for interpolatory braking of the single axis (SW 6.4 and higher) | |

## 7.3 Setting data

| Number | Names | Name | Reference |
|---|---|---|---|
| **Axisspecific ($SA_ ... )** | | | |
| 43600 | IPOBRAKE_BLOCK_EXCHANGE | Braking ramp block change condition SW 6.2 and higher | |
| 43610 | ADISPOSA_VALUE | Braking ramp tolerance window with SW 6.4 and higher | |

## 7.4 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, “Diagnostics Guide”
and the online help of MMC 101/102/103 systems.

■

# SINUMERIK 840D/840Di/810D
# Description of Functions
# Extended Functions (FB2)

# Oscillation (P5)

■

# 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 beyond block limits.
  Any other axes can interpolate during the oscillation motion. 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.
  The infeed is possible simultaneously on several axes. 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 the programmed position is reached 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, MMC, handwheel or directional keys.

– The feedrate velocity of the oscillation axis can be altered through a value input in the NC program, PLC, MMC or via an override. The feedrate can be programmed to be dependent on a master spindle, rotary axis or spindle (revolutional feedrate).

**References:** /FB/; V1, “Feedrates”

– The oscillation movement can be controlled entirely by the PLC.

■

# Detailed Description 2

**Methods of oscillation control**

There are two modes of oscillation:

1. Asynchronous oscillation,
   which is active beyond block limits and can also be started from PLC/MMC,

   and

2. Oscillation as controlled by motion-synchronous actions.
   In this case, asynchronous oscillation and an infeed movement are coupled with one another 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.

## 2.1 Asynchronous oscillation

**Properties**

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 VDI interface that the axis reacts in the desired way to VDI signals. These signals include program end, operating mode changeover and single block.
- The oscillation axis can act as the input axis for the transformations (e.g. inclined axis).

**References:** /FB/, M1, “Transmit/Peripheral Surface Transformation”

- The oscillation axis can act as the master axis for gantry and coupled motion axes.

**References:** /FB/, G1, “Gantry Axis”

- 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 parts program.
- The oscillation movement can likewise be started, influenced and stopped from the PLC/MMC.
- Interpolatory oscillation is not possible (e.g. oblique oscillation).

### 2.1.1 Influences on asynchronous oscillation

**Setting data**

The setting data required for oscillation can be set with special language commands in the NCK parts program, via the HMI and/or the PLC.

**Feed velocity**

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 positioning axes).
- 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 parts program and setting data whether the changed velocity must take effect immediately or whether it should be activated at the next reversal point.
- The feed velocity can be influenced via the override (axial VDI signal and programmable).
- If Dry run is active, the dry run velocity setting is applied if it is higher than the currently programmed velocity. (SW version 5 and earlier).
  From SW version 6, the effectiveness of the dry run feedrate set in SD 42100: DRY_RUN_FEED can be controlled with SD 42101: DRY_RUN_FEED_MODE. See
  **References:** /FB1/, V1, Feedrates
- Velocity overlay/path overlay can be influenced by the handwheel.
  See also Table 2-1.

**References:** /FB/, H1, “Manual and Handwheel Travel”

- The oscillation axis can be moved with revolutional feedrate.

**Revolutional feedrate**

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 traverse (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 VDI interface signal *"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 each reversal point.
The setting can be altered in the following blocks of the NC program. It is then effective in block synchronism from the next applicable reversal point.
The stop time can be altered 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 2-1 Effect of stop time

| Stop time setting | Procedure |
|---|---|
| –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 exact stop fine 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 movement towards the other reversal point and back. See table:

---

**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 2-2 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 sparkingout strokes | Number of sparking-out strokes is not equal to 0, no-end position active | After appropriate reversal point is reached, the number of sparking-out strokes specified in command is processed. |
| Deactivation with sparking-out strokes and defined end position (optional) | Number of sparking-out strokes is not equal to 0, end position active | After appropriate reversal point is reached, the number of sparking-out strokes specified in command is processed followed by approach to specified end position. |
| Deactivation without sparkingout strokes, but with defined end position (optional) | Number of sparking-out strokes equals 0, end position active | After appropriate reversal point is reached, axis is traversed to specified end position. |

**NC language**

The NC programming language allows asynchronous oscillation to be controlled from the parts program. The following functions allow asynchronous oscillation to be activated and controlled as a function of NC program execution.

---

**Note**

If the setting data are directly written in the parts program, then the data change takes effect prematurely with respect to processing of the parts program (at the preprocessing time). It is possible to re-synchronize the parts program and the oscillation function commands by means of a preprocessing stop (STOPRE).

---

**References:** /PA/, 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 a positioning axis again and can then be used normally.
  If an axis is to be used for oscillation, then it must be enabled beforehand with a WAITP(axis) call.
  This also applies if oscillation must be 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. With SW version 3.2 and higher 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. **Stop 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 feedrate.

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 revolutional feedrate, the corresponding dependencies must be indicated as described in Description of Functions V1.

6. **Setting control settings for sequence of movements:**

- OSCTRL[oscillation axis] = (set options, reset options)

  The set options are defined as follows (the reset options deselect the settings):

Table 2-3 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 set implicitly according to Table 2-3, such 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 transferred to the appropriate setting data 43790: OSCILL_START_POS in synchronism with the block in the main run and remains valid until the data is next changed. Bit 9 in setting data 43770 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 43790: 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 examples in 6.4.3.

**Programming example**

Chapter 6 gives an example containing all the important elements for asynchronous oscillation.

## 2.1.2 Asynchronous oscillation under PLC control

**Activation**

The function can be selected from the PLC via setting data OSCILL_IS_ACTIVE in all operating modes except for MDA Ref and JOG Ref.

**Settings**

The following criteria can be controlled from the PLC via setting data: Activation and deactivation of oscillation movement, positions of reversal points 1 and 2, 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 MMC 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.

Axes always react to the stop bits at the VDI interface IS "Stop" (DB31, ... DBX28.6) and IS "Stop at next reversal point" (DB31, ... DBX28.5) regardless of whether or not they are operating under the control of the PLC.

### 2.1.3 Special reactions during asynchronous oscillation

**With PLC control**

The PLC program can take over the control of an oscillation axis via VDI signals. These VDI signals also include program end, operating mode changeover and single block.

The following VDI interface signals are ignored in SW 6.2 and earlier: Feed/spindle stop and NC STOP; the resulting deceleration request is suppressed in the case of delete distance-to-go.

In SW 6.3 and later, an asynchronous reciprocating axis interpolated by the main run reacts to NC STOP, alarm handling, end of program, program control and RESET, independently of the NC program.

The PLC controls the axis/spindle via the axial VDI interface (PLC→NCK) by means of IS “PLC controls axis” (DB31, ... DBX28.7) == 1

For further information about axes with PLC control, please see:
**References:** /FB/, P2, “Positioning Axes”

**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**

If the oscillation axis is not under PLC control, it is stopped by means of a braking ramp.

**With PLC control**

In this case, deceleration is suppressed and must be initiated by the PLC.
The following applies to **both** cases: After the axis has been stopped, the appropriate reversal point is approached (see OSCILL_CONTROL_MASK, Chapter 4) and the distance-to-go deleted. The sparking-out strokes are then executed and the end position approached if this has been set such in OSCILL_CONTROL_MASK.
The oscillation motion is then finished.

---

**Note**

During grinding, the calipers can be put into action via axial delete distance-to-go.

---

**EMERGENCY STOP**

In the event of an EMERGENCY STOP, the axis is decelerated by the servo (by cancellation of servo enable and follow-up). The oscillation movement is thus terminated and must be restarted if necessary.

**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).

## Special reactions during asynchronous oscillation (continued)

**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 the oscillation axis violates 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 output.

---

**Caution**

Protection zones are not effective!

---

**Follow up operation**

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 beyond block limits).

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 settings apply: The type of movement first started has priority.

Table 2-4 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:

**VDI 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 the case of a block search, the last valid oscillation function is registered and is activated – depending on machine data OSCILL_MODE_MASK – either immediately after NC start (on approach to approach position after block search) or after the approach position has been reached after block search (default setting).
OSCILL_MODE_MASK Bit 0:

| | |
|---|---|
| 0 | Oscillation starts after approach position is reached |
| 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.

## 2.2 Oscillation controlled by synchronized actions

**Principle**

An asynchronous oscillation movement is coupled via synchronized actions with an infeed motion and controlled accordingly.

**References:** /FB/, S5, "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 reversal point (see 2.2.1).
2. Infeed in reversal point range (see 2.2.2).
3. Infeed at both reversal points (see 2.2.3).
4. Stopping oscillation movement at reversal point until infeed is terminated (see 2.2.4).
5. Enable oscillation movement (see 2.2.5).
6. Preventing premature start of partial infeed (see 2.2.6).

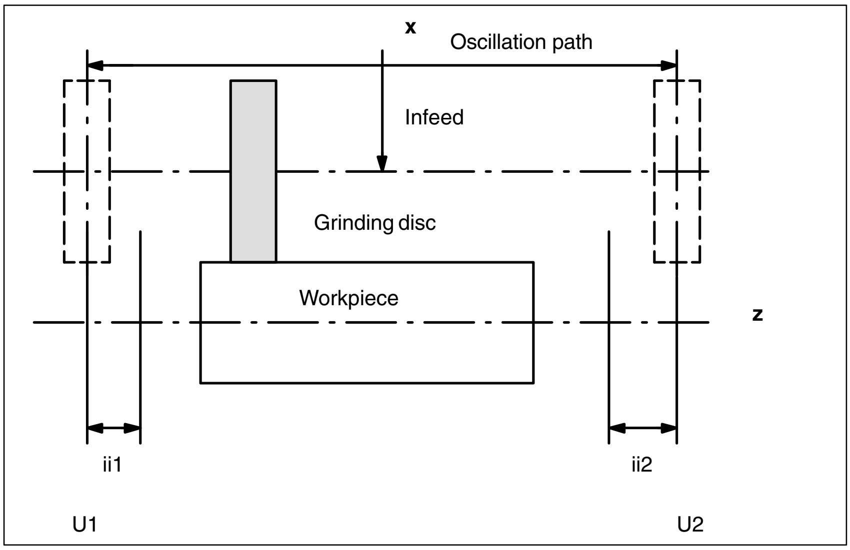



Fig. 2-1 Arrangement of oscillation and infeed axes plus terms

| | | |
|---|---|---|
| Legend: | U1 | Reversal point 1 |
| | U2 | Reversal point 2 |
| | ii1 | Reversal range 1 |
| | ii2 | Reversal range 2 |

**Programming**

The parameters governing oscillation (see 2.2.7) must be defined before the movement block containing the assignment between the infeed and oscillation axes (see 2.1.1), the infeed definition (POSP) and the motion-synchronous actions:

The oscillation axis is enabled via a WAITP[oscillation axis] (see MD $MA_AUTO_GET_TYPE), allowing the oscillation parameters to be transferred, i.e. into the setting data, simultaneously. The symbolic names, e.g. $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. Subsequent changes to setting data are irrelevant in this respect.
For motion-synchronous actions with $$AA_REVERSE_POS values, the comparison values apply within the **interpolation**. 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 elements which have not yet been discussed are explained in more detail in the following sections.
Some examples are described in Chapter 6.

---

**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**

With SW 3.2 and higher, 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:** /FB/, S5, “Motion-synchronous actions”.

Example 1

Oscillation, reversal position firmly set via setting data:

```
$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       ; Oscillation ON
OS[Z]=0                            ; Oscillation OFF

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.

```
$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       ; Oscillation ON
OS[Z]=0                            ; Oscillation OFF

M30
```

## 2.2.1 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] | Actual position of oscillation axis Z in machine coordinate system |
| $SA_OSCILL_REVERSE_POS1[Z] | Position of reversal point 1 of oscillation axis |
| $AA_OVR[X] | Axial override of infeed axis |
| $AA_OVR[Z] | Axial override of oscillation axis |

Explanation of vocabulary words:

| | |
|---|---|
| WHENEVER ... DO ... | Whenever condition is fulfilled, then... |

**Infeed**

The absolute infeed value is defined by instruction POSP.
See 2.2.8.

**Assignment**

The assignment between the oscillation axis and the infeed axis is defined by instruction OSCILL.
See 2.2.7.

## 2.2.2 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 variable 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 Fig. 2-1.

**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] | Actual position of oscillation axis Z |
| $SA_OSCILL_REVERSE_POS1[Z] | Position of reversal point 1 of oscillation axis |
| $AA_OVR[X] | Axial override of infeed axis |
| ii1 | Size of reversal range (user variable) |

Explanation of vocabulary words:

| | |
|---|---|
| WHENEVER ... DO ... | Whenever 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 point 2 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 Fig. 2-1.

**Programming**

**Reversal point range 2:**
WHENEVER $AA_IM[Z] < $SA_OSCILL_REVERSE_POS2[Z] – ii2 DO
$AA_OVR[X] = 0

Explanation of system variables:

| | |
|---|---|
| $AA_IM[Z] | Actual position of oscillation axis Z |
| $SA_OSCILL_REVERSE_POS2[Z] | Position of reversal point 2 of oscillation axis |
| $AA_OVR[X] | Axial override of infeed axis |
| ii2 | Size of reversing point range 2 (user variable) |

**Infeed**

The absolute infeed value is defined by instruction POSP.
See Subsection 2.2.8 .

**Assignment**

The assignment between the oscillation axis and the infeed axis is defined by instruction OSCILL.
See Subsection 2.2.7.

### 2.2.3 Infeed at both reversal points

**Principle**

The functions described above for infeed at the reversal point and in the reversal point range can be freely combined.

**Combinations**

Infeed:

| | |
|---|---|
| an U1 | at U2 |
| at U1 | range U2 |
| range U1 | at U2 |
| range U1 | range U2 |

**One-sided infeed**

at U1
at U2
range U1
range U2

These options are described in Subsections 2.2.1 and 2.2.2.

## 2.2.4 Stopping oscillation movement at 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 part infeed has been executed. This synchronized action can be omitted if the oscillation axis need not wait at reversal point 1 until part 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 oscillation axis |
| $AA_OVR[oscillation axis] | Axial override of oscillation axis |
| $AA_OVR[infeed axis] | Axial override of infeed axis |

**Function**

**Reversal point 2:**
Every time the oscillation axis reaches reversal position 2, it must be stopped by means of a 0 override and the infeed movement started.

**Application**

The synchronized action is used to hold the oscillation axis stationary until part infeed has been executed. This synchronized action can be omitted if the oscillation axis need not wait at reversal point 2 until part 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 of system variables:

| | |
|---|---|
| $AA_IM[oscillation axis] | Current position of oscillation axis |
| $SA_OSCILL_REVERSE_POS2[oscillation axis] | Reversal point 2 of oscillation axis |
| $AA_OVR[oscillation axis] | Axial override of oscillation axis |
| $AA_OVR[infeed axis] | Axial override of infeed axis |

## 2.2.5 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. part 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 distance-to-go for infeed axis in workpiece coordinate system<br>Path section of infeed axis |
| $AA_OVR[oscillation axis] | Axial override for oscillation axis |

Explanation of vocabulary words:

| | |
|---|---|
| WHENEVER ... DO ... | Whenever condition is fulfilled, then... |

## 2.2.6 Prevent premature starting of partial infeed

**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 example, 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 distance-to-go for infeed axis in workpiece coordinate system<br>Path section of infeed axis |
| $AC_MARKER[index] | Channel-specific flag with index |
| $AA_IM[oscillation axis] | Current position of oscillation axis |
| $SA_OSCILL_REVERSE_POS1[oscillation axis] | Reversal point 1 of oscillation axis |
| $AA_OVR[infeed axis] | Axial override for infeed axis |

Explanation of vocabulary words:

| | |
|---|---|
| WHENEVER ... DO ... | Whenever condition is fulfilled, then... |

### 2.2.7 Assignment of oscillation and infeed axes OSCILL

**Function**

One or several infeed axes are assigned to the oscillation axis with command OSCILL. The oscillation movement is started.
The PLC is informed of which axes have been assigned via the VDI 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.

### 2.2.8 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 control 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 infeed axis after all partial infeeds have been executed. |
| Partial section | Part infeed at reversal point/reversal point range |
| Mode | 0<br>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<br>The part length is adjusted such that the total of all calculated part lengths corresponds exactly to the path up to the target point. |

### 2.2.9 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 current oscillation motion is braked and the axis then traversed in the opposite direction in response to edge-triggered PLC input signal **Oscillation reversal** (DB31 DBB28 bit0). The braking operation is checked back via PLC output signal **Oscillation reversal active** (DB31 DBB100 bit 2).

The braking position of the axis can be accepted as the **new reversal position** by means of PLC signal **Change reversal position** (DB31 DBB28 Bit4).

The PLC input signal **Select reversal position** (DB31 DBB28 bit 3) is ignored provided that the change is made in relation to the last issued *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.

**Stop time, exact stop**

No stop time is applied to direction changes initiated by an *External oscillation reversal* command. The axis waits for the exact stop fine signal. Any configured exact stop condition is fulfilled.

**Infeed movement**

With non-modal oscillation, no infeed motion is performed when the direction is changed by an *External oscillation reversal* command, as the reversal position is not reached and the corresponding synchronized action cannot therefore be 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 examples.

**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 6.5.1.

■

# Supplementary Conditions 3

**Availability of the "Oscillation" function**

Oscillation is an option with order number 6FC5 251-0AB04-0BA0.
Asynchronous and modal oscillation is available with SW 2 and higher for NCU570, 571, 572, 573.
Oscillation with motion-synchronous actions is available with NCU 572 and 573.

■

# Data Descriptions (MD, SD) 4

## 4.1 Machine data

| | | | |
|---|---|---|---|
| **11460**<br>MD number | **OSCILL_MODE_MASK**<br>Mode screen form for asynchronous oscillation | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 0xFFFF | |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – | |
| Data type: DWORD | Applies from SW: 2.0 | | |
| Meaning: | Bit 0<br>Value 1 In the case of block search, the oscillation movement is started immediately after NC start, i.e. during approach to the approach position, provided it has been activated in the program section being processed.<br>Value 0 The oscillation movement is not started until the approach position is reached. | | |

## 4.2 Setting data

### Axis/spindle specific data

| | | |
|---|---|---|
| **43700**<br>SD number | **OSCILL_REVERSE_POS1[axis]**<br>Oscillation reversal point 1 | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective after: immediately | | Unit: mm, degrees |
| Data type: DOUBLE | Applies from SW: 2.0 | |
| Meaning: | Position of oscillation axis at reversal point 1 | |
| Application example(s) | NC language: OSP1[axis]=position | |
| Related to .... | OSCILL_REVERSE_POS2<br>MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | |

| | | |
|---|---|---|
| **43710**<br>SD number | **OSCILL_REVERSE_POS2[axis]**<br>Oscillation reversal point 2 | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** |
| Changes effective after: immediately | | Unit: mm, degrees |
| Data type: DOUBLE | Applies from SW: 2.0 | |
| Meaning: | Position of oscillation axis at reversal point 2 | |
| Application example(s) | NC language: OSP2[axis]=position | |
| Related to .... | OSCILL_REVERSE_POS1<br>MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | |

| | | |
|---|---|---|
| **43720**<br>SD number | **OSCILL_DWELL_TIME1[axis]**<br>Stop time at oscillation reversal point 1 | |
| Default setting: 0 | Minimum input limit: –2 | Maximum input limit: *** |
| Changes effective after: immediately | | Unit: s |
| Data type: DOUBLE | Applies from SW: 2.0 | |
| Meaning: | Stop time of oscillation axis at reversal point 1 | |
| Application example(s) | NC language: OST1[axis]=time | |
| Related to .... | OSCILL_DWELL_TIME2<br>MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | |

| | | |
|---|---|---|
| **43730**<br>SD number | **OSCILL_DWELL_TIME2[axis]**<br>Stop time at oscillation reversal point 2 | |
| Default setting: 0 | Minimum input limit: –2 | Maximum input limit: *** |
| Changes effective after: immediately | | Unit: s |
| Data type: DOUBLE | Applies from SW: 2.0 | |
| Meaning: | Stop time of oscillation axis at reversal point 2 | |
| Application example(s) | NC language: OST2[axis]=time | |
| Related to .... | OSCILL_DWELL_TIME1<br>MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | |

| | | | |
|---|---|---|---|
| **43740**<br>SD number | **OSCILL_VELO[axis]**<br>Feed velocity of oscillation axis | | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** | |
| Changes effective after: immediately | | | Unit: mm/min<br>rev/min |
| Data type: DOUBLE | | Applies from SW: 2.0 | |
| Meaning: | Feed velocity of oscillation axis | | |
| Application example(s) | NC language: FA[axis]=Fvalue | | |
| Related to .... | MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | | |

| | | | |
|---|---|---|---|
| **43750**<br>SD number | **OSCILL_NUM_SPARK_CYCLES[axis]**<br>Number of sparking-out strokes | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: *** | |
| Changes effective after: immediately | | | Unit: 1 |
| Data type: DWORD | | Applies from SW: 2.0 | |
| Meaning: | Number of sparking-out strokes which are executed on completion of oscillation movement | | |
| Application example(s) | NC language: OSNSC[axis]=number of strokes | | |
| Related to .... | MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | | |

| | | | |
|---|---|---|---|
| **43760**<br>SD number | **OSCILL_END_POS[axis]**<br>End position of oscillation axis | | |
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** | |
| Changes effective after: immediately | | | Unit: mm, degrees |
| Data type: DOUBLE | | Applies from SW: 2.0 | |
| Meaning: | Position to be approached by oscillation axis after execution of sparking-out strokes. | | |
| Application example(s) | NC language: OSE[axis]=position | | |
| Related to .... | MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | | |

| | | | |
|---|---|---|---|
| **43770**<br>SD number | **OSCILL_CTRL_MASK[axis]**<br>Oscillation sequence control screen form | | |
| Default setting: 0 | Minimum input limit: – | Maximum input limit: – | |
| Changes effective after: immediately | | | Unit: – |
| Data type: BYTE | | Applies from SW: 2.0 | |
| Meaning: | Bit screen form, see following Table 4-1 | | |
| Application example(s) | NC language: OSCTRL[axis]=(setting options, resetting options) | | |
| Related to .... | MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | | |

| | | | |
|---|---|---|---|
| **43780**<br>SD number | **OSCILL_IS_ACTIVE[axis]**<br>Activation/deactivation of oscillation motion | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after: immediately | | | Unit: – |
| Data type: BOOLEAN | | Applies from SW: 2.0 | |
| Meaning: | Activation/deactivation of oscillation motion<br><br>Note:<br>The MD 10710: $MN_PROG_SD_RESET_SAVE_TAB can be set such that the value written from the part program during reset is accepted by the active file system (i.e. the value is retained beyond the reset) | | |
| Application example(s) | NC language: OS[axis]=1<br>OS[axis]=0 | | |
| Related to .... | MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | | |

| 43790<br>SD number | OSCILL_START_POS[axis]<br>Start position of oscillation axis | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** | |
| Changes effective after: immediately | | | Unit: mm, degrees |
| Data type: DOUBLE | | Applies from SW: 7.2 | |
| Meaning: | Position approached by the oscillation axis when oscillation commences if this has been set in<br>$SA_OSCILL_CTRL_MASK.<br>Note:<br>MD 10710: $MN_PROG_SD_RESET_SAVE_TAB can be set<br>such that the value written from the parts program during reset is accepted by the active file system (i.e. the value is retained beyond the reset). | | |
| Application example(s) | NC language: | OSB[axis]=position<br>OSB[axis]=position | |
| Related to .... | MD 10710 $MN_PROG_SD_RESET_SAVE_TAB | | |

Table 4-1 Bit significance in screen form OSCILL_CTRL_MASK

| Bit no. | Meaning in OSCILL_CTRL_MASK |
|---|---|
| 0.1 | 0: Stop at next reversal point on deactivation of oscillation movement<br>1: Stop at reversal point 1 on deactivation of oscillation movement<br>2: Stop at reversal point 2 on deactivation of oscillation movement<br>3: On deactivation of oscillation movement, do not approach reversal point unless sparking-out strokes are programmed |
| 2 | 1: Approach end position after next sparking-out |
| 3 | 1: 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) |
| 4 | 1: If the oscillation movement is aborted by delete distance-to-go, then the appropriate reversal position is approached as for deactivation |
| 5 | 1: New feedrate setting not effective until the next reversal point |
| 6 | 1: If feedrate setting is 0, path overlay is active, or otherwise velocity overlay |
| 7 | 1: For rotary axes DC (shortest path) |
| 8 | Execute sparking-out stroke as single stroke, not as double stroke |
| 9 | Approach start position when oscillation commences |

■

# 5 Signal Descriptions

**VDI input signals**

The PLC user program uses the following signals to control the oscillation process.

| DB31, ...<br>**DBX28.0**<br>Data Block | **External oscillation reversal**<br><br>Signal(s) to axis/spindle | |
|---|---|---|
| Edge evaluation: yes | Signal(s) updated: Cyclic | Signal(s) valid from SW: 7.2 |
| Signal state 1 or signal transition 0 ——> 1 | Brake oscillation motion and move oscillation axis in the opposite direction. | |
| Signal state 0 or signal transition 1 ——> 0 | Continue oscillation without interruption | |

| DB31, ...<br>**DBX28.3**<br>Data Block | **Set reversal point**<br><br>Signal(s) to axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.0 |
| Signal state 1 or signal transition 0 ——> 1 | Reversal point 2 | |
| Signal state 0 or signal transition 1 ——> 0 | Reversal point 1 | |

| DB31, ...<br>**DBX28.4**<br>Data Block | **Alter reversal point**<br><br>Signal(s) to axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.0 |
| Signal state 1 or signal transition 0 ——> 1 | The selected reversal point can be altered by manual traverse.<br>In conjunction with DB31, ...DBX28.0:<br>The position at which axis is braked after external oscillation reversal must be accepted as new reversal point. | |
| Signal state 0 or signal transition 1 ——> 0 | The selected reversal point cannot be altered by manual traverse.<br>In conjunction with DB31, ...DBX28.0:<br>No change to reversal point | |
| Related to .... | DBX28.3 | |

| DB31, ...<br>**DBX28.5**<br>Data Block | **Stop at next reversal point**<br><br>Signal(s) to axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.0 |
| Signal state 1 or signal transition 0 ——> 1 | The oscillation movement is interrupted at the next reversal point. | |
| Signal state 0 or signal transition 1 ——> 0 | The oscillation movement continues after the next reversal point. | |
| Related to .... | DBX28.6, DBX28.7 | |

| DB31, ...<br>**DBX28.6**<br>Data Block | **Stop along braking ramp**<br><br>Signal(s) to axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.0 |
| Signal state 1 or signal transition 0 ——> 1 | The axis is decelerated along a ramp, the oscillation movement is interrupted. | |
| Signal state 0 or signal transition 1 ——> 0 | The oscillation movement continues without interruption. | |
| Related to .... | DBX28.5, DBX28.7 | |

| DB31, ...<br>**DBX28.7**<br>Data Block | **PLC controls axis**<br><br>Signal(s) to axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.0 |
| Signal state 1 or signal transition 0 ——> 1 | Axis is controlled by the PLC.<br>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 signal transition 1 ——> 0 | Axis is not controlled by the PLC. | |
| Related to .... | DBX28.5, DBX28.6 | |

## VDI output signals

The NCK makes the following signals available to the PLC user program.

| DB31, ...<br>**DBX100.2**<br>Data Block | **Oscillation reversal active**<br><br>Signal(s) from axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 7.2 |
| Signal state 1 or signal transition 0 ——> 1 | The deceleration period after external oscillation reversal (DB31, ...DBX28.0) is active | |
| Signal state 0 or signal transition 1 ——> 0 | No deceleration after external oscillation reversal is active | |

| DB31, ...<br>**DBX100.3**<br>Data Block | **Oscillation cannot start**<br><br>Signal(s) from axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 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 signal transition 1 ——> 0 | The oscillation movement can be started. | |

| DB31, ...<br>**DBX100.4**<br>Data Block | **Error during oscillation movement**<br><br>Signal(s) | |
|---|---|---|
| Edge evaluation: | Signal(s) updated: | Signal(s) valid from SW: 2..1 |
| Signal state 1 or signal transition 0 ——> 1 | The oscillation movement has been aborted. | |
| Signal state 0 or signal transition 1 ——> 0 | The oscillation movement is being executed correctly. | |

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX100.5**<br>Data Block | **Sparking-out active**<br><br>Signal(s) from axis/spindle | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ——> 1 | The axis is executing sparking-out strokes. | | |
| Signal state 0 or signal transition 1 ——> 0 | The axis is not currently executing sparking-out strokes. | | |
| Related to .... | DBX100.7 | | |

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX100.6**<br>Data Block | **Oscillation movement active**<br><br>Signal(s) from axis/spindle | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ——> 1 | The axis is executing an oscillation movement between 2 reversal points. | | |
| Signal state 0 or signal transition 1 ——> 0 | The axis is not currently oscillating. | | |
| Signal irrelevant for ... ... | DBX100.7 = 0 | | |
| Related to .... | DBX100.7 | | |

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX100.7**<br>Data Block | **Oscillation active**<br><br>Signal(s) from axis/spindle | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ——> 1 | The axis is currently being traversed as an oscillation axis. | | |
| Signal state 0 or signal transition 1 ——> 0 | The axis is a positioning axis. | | |
| Related to .... | DBX100.5, DBX100.6 | | |

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX104.0 - 7**<br>Data Block | **Active infeed axes**<br><br>Signal(s) from axis/spindle | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 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 signal transition 1 ——> 0 | The associated axis is not an infeed axis. | | |
| Related to .... | DBX100.7 | | |

■

# Notes

# Examples 6

**Prerequisites**

The examples given below require components of the NC language specified in the sections entitled:

– Asynchronous oscillation

**and**

– Oscillation controlled by synchronized actions.

## 6.1 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 extract**

```
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         ; Feedrate oscillation axis
OSNSC[Z]=3         ; Three sparking-out strokes
OSE[Z]=30          ; End position
OS[Z]=1 F500 X15   ; Start oscillation, infeed X axis
                   ; with feedrate 500, infeed target 15
```

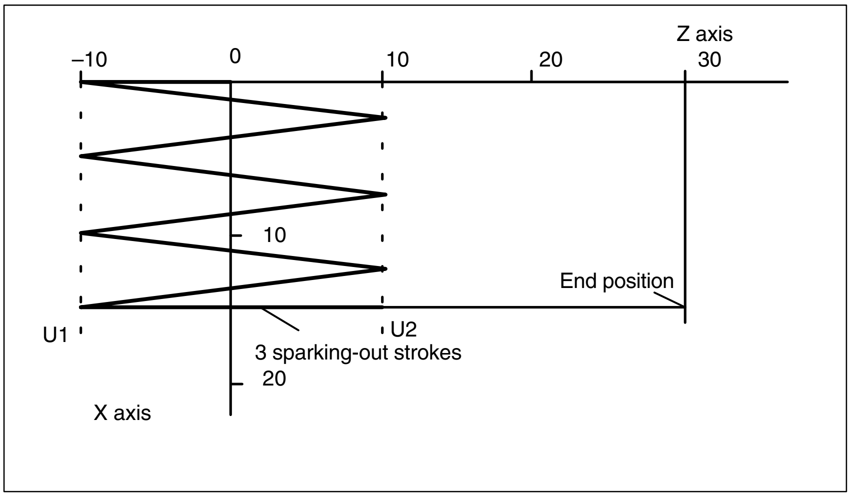



Fig. 6-1 Sequences of oscillation movements and infeed, example 1

## 6.2 Example 1 of oscillation with synchronized actions

**Task**

Direct infeed must take place at reversal point 1; the oscillation axis must wait until the part 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 part infeed has been executed. Axis Z is the oscillation axis and axis X the infeed axis. (See 2.2).

---

**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 extract**

```
;  Example 1: Oscillation with synchronized actions

OSP1[Z]=10 OSP2[Z]=60      ; Declare 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       ; Feedrate for oscillation axis, feedrate
                           ; for infeed axis
OSCTRL[Z]=(1+8+16,0)       ; Deactivate oscillation movement at reversal point
                           ; 1
                           ; Sparking-out after deletion of distance-to-go and
                           ; approach end position
                           ; Approach programmed reversal
                           ; point after deletion of distance-to-go
OSNSC[Z]=3                 ; 3 sparking-out strokes
OSE[Z]=0                   ; End position = 0;
WAITP(Z)                   ; Enable oscillation for Z axis

; Motion-synchronous actions:
;
;Whenever          the current position of the oscillation axis in the machine coordinate system
; is not equal to  reversal position 1,
; then             set the flag with index 1 to a value of 0 (reset flag 1)
WHENEVER $AA_IM[Z]<>$SA_OSCILL_REVERSE_POS1[Z] DO $AC_MARKER[1]=0
;
;Whenever          the current position of the oscillation axis in the machine coordinate system
; is lower than    the beginning of reversal point range 2 (here: Reversal point 2 – 6),
; then             set the axial override of the infeed axis to 0%
; and              set flag with index 2 to value 0 (reset flag 2).
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]–6 DO $AA_OVR[X]=0 $AC_MARKER[2]=0
;
;Whenever          the current position of the oscillation axis in the machine coordinate system
; is the same as   reversal position 1,
; then             set the axial override of oscillation axis to 0%
; and              set the axial override of infeed axis to 100% (i.e. to cancel the
;                  preceding synchronized action!)
WHENEVER $AA_IM[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[Z]=0 $AA_OVR[X]=100
;

;Whenever          the distance-to-go of the part infeed
; is equal to      0,
; then             set flag with index 2 to a value of 1
; and              set flag with index 1 to a value of 1
```

```
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[2]=1 $AC_MARKER[1]=1
;
;Whenever        the flag with index 2
; is equal to     1,
; then            set the axial override of the infeed axis to 0% to prevent premature infeed
;                 (oscillation axis has not yet exited from reversal
;                 position 1).
WHENEVER $AC_MARKER[2]==1 DO $AA_OVR[X]=0
;
;Whenever        the flag with index 1
; is equal to     1,
; then            set the axial override of the infeed axis to 0% to prevent premature
;                 infeed (oscillation axis has not yet exited from reversal point range 2)
; and             set axial override of oscillation axis to 100% ('start' oscillation).
WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
;

;If the current position of the oscillation axis in the machine coordinate system
; is equal to     reversal position 1,
; then            set the axial override of the oscillation axis to 100%
; and             set the axial override of the infeed axis to 0% (in order
;                 to cancel the second synchronized action 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 as infeed axis
                                 ; to oscillation axis Z; axis X must
                                 ; infeed to end position 5 in part steps of
                                 ; 1 and the total of all part lengths
                                 ; must correspond exactly to the end position
;
M30                              ; Program end
```

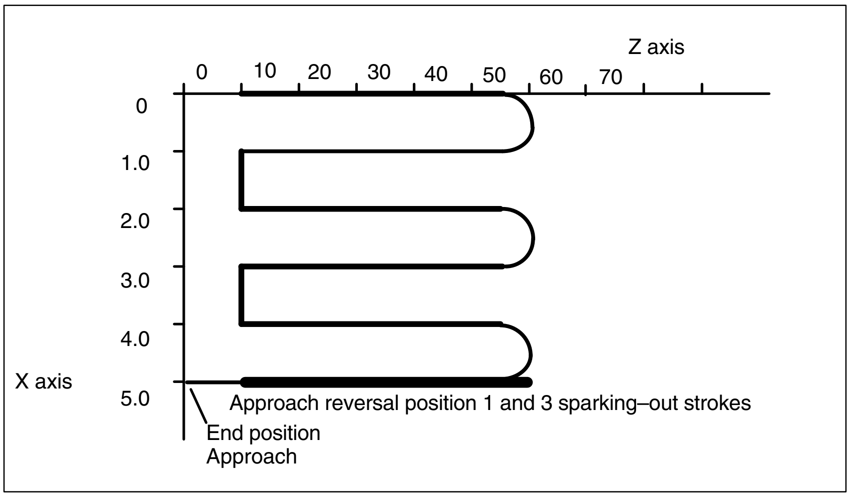



Fig. 6-2 Sequences of oscillation movements and infeed, example 1

## 6.3 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 part infeed has been executed. Axis Z is the oscillation axis and axis X the infeed axis.

**Program section**

Example 2: Oscillation with synchronized actions

```
DEF INT ii2                 ; Define variables for reversal point range 2
;
OSP1[Z]=10 OSP2[Z]=60       ; Declare 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        ; Feedrate for oscillation axis, feedrate for infeed axis
OSCTRL[Z]=(2+8+16,1)        ; Deactivate oscillation movement at reversal point 2
                            ; After deletion of distance-to-go sparking-out and approach end position
                            ; After deletion of distance-to-go approach appropriate reversal position
OSNSC[Z]=3                  ; 3 sparking-out strokes
OSE[Z]=70                   ; End position = 70;
ii2=2                       ; Set reversal point range
WAITP(Z)                    ; Enable oscillation for Z axis

; Motion-synchronous actions:
; Whenever                  the current position of the oscillation axis in the machine coordinate system
; is lower than             the start of reversal point range 2,
; then                      set the axial override of the infeed axis to 0%
; and                       set the flag with index 0 to a value of 0
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]–ii2 DO $AA_OVR[X]=0 $AC_MARKER[0]=0
;
; Whenever                  the current position of the oscillation axis in the machine coordinate system
; is equal to or greater than  reversal position 2,
; then                      set the axial override of the oscillation axis to 0%
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0
;
; Whenever                  the distance-to-go of the part infeed
; is equal to               0,
; then                      set the flag with index 0 to a value of 1
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1
;
; Whenever                  the flag with index 0
; is equal to               1,
; then                      set the axial override of the infeed axis to 0%
;                           in order to prevent premature infeed (oscillation axis has not yet exited
;                           from reversal point range 2, infeed axis is ready to infeed again)
; and                       set the axial override of the oscillation axis to 100% (thus canceling
;                           the 2nd synchronized action)
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
;
OSCILL[Z]=(X) POSP[X]=(5,1,1)        ; Start axes
                                     ; Axis X is assigned to oscillation axis Z
                                     ; as the infeed axis
                                     ; Axis X must traverse to end position
                                     ; 5 in steps of 1
```

;
M30

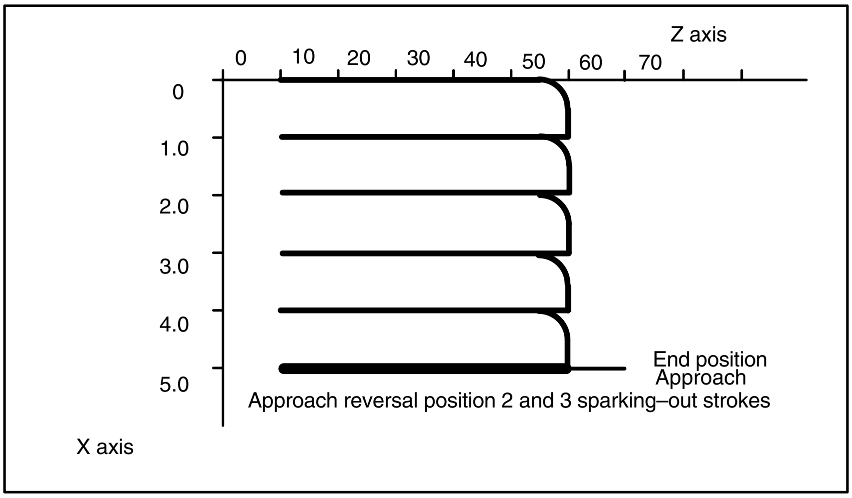



Fig. 6-3 Sequences of oscillation movements and infeed, example 2

# 6.4 Examples for starting position

## 6.4.1 Define starting position via language command

```
WAITP(Z)                    ; Allow oscillation for Z axis
OSP1[Z]=10 OSP2[Z]=60       ; Declare 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       ; Feedrate for oscillation axis,
                            ; Feedrate for infeed axis
OSCTRL[Z]=(1+8+16,0)        ; Deactivate oscillation movement at
                            ; reversal point 1
                            ; Sparking-out after deletion
                            ; of distance-to-go and approach
                            ; end position, Approach programmed
                            ; reversal point after deletion of
                            ; distance-to-go
OSNSC[Z]=3                  ; 3 sparking-out strokes
OSE[Z]=0                    ; End position = 0
OSB[Z]=0                    ; Starting position = 0
OS[Z]=1 X15 F500            ; Start oscillation, continuous infeed
OS[Z]=0                     ; Deactivate oscillation
WAITP(Z)                    ; Wait for end of 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.

### 6.4.2 Start oscillation via setting data

```
WAITP(Z)
STOPRE
$SA_OSCILL_REVERSE_POS1[ Z ] = –10   ; Reversal point 1 = –10
$SA_OSCILL_REVERSE_POS2[ Z ] = 30    ; Reversal point 2 = 30
$SA_OSCILL_START_POS[Z] = –50        ; Starting position = –50
$SA_OSCILL_CTRL_MASK[Z] = 512        ; Approach starting position,
                                     ; When deactivated, stop at
                                     ; next reversal point
                                     ; do not approach end position
                                     ; No sparking out strokes with
                                     ; deletion of distance to go
$SA_OSCILL_VELO[ Z ] = 5000          ; Feedrate for oscillation axis
$SA_OSCILL_IS_ACTIVE[ Z ] = 1        ; Start
$SA_OSCILL_DWELL_TIME1[ Z ] = –2     ; without waiting for exact stop
$SA_OSCILL_DWELL_TIME2[ Z ] = 0      ; Wait for exact stop fine
STOPRE
X30 F100
$SA_OSCILL_IS_ACTIVE[ Z ] = 0        ; Stop
WAITP(Z)
M30
```

**Explanations**

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.

### 6.4.3 Non-modal oscillation (starting position = reversal point 1)

**Oscillation with synchronized actions**

```
N701 ;  Oscillation with synchronized actions, starting position == reversal
        point 1
  ;
N702 OSP1[Z]=10 OSP2[Z]=60      ; Declare 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      ; Feedrate for oscillation axis,
                                ; Feedrate for infeed axis
N705 OSCTRL[Z]=(1+8+16.0)       ; Deactivate oscillation motion at
                                ; reversal point 1
                                ; Sparking-out after deletion of
                                ; distance-to-go and approach end position
                                ; Approach programmed reversal
                                ; point after deletion of distance-to-go
N706 OSNSC[Z]=3                 ; 3 sparking-out strokes
N707 OSE[Z]=0                   ; End position = 0
N708 OSB[Z]=10                  ; Starting position = 10
N709 WAITP(Z)                   ;Enable oscillation for Z axis
;
_____________________________________________
; Motion-synchronous actions:
; Set marker with index 2 to 1 (initialization)
WHEN TRUE DO $AC_MARKER[2]=1

; Whenever marker with index 2 equals 0 and
; the current position of oscillation axis is not equal to reversal point 1,
; then
; set 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

; Whenever
; current position of oscillation axis is less than the start of
; reversal area 2,
; then
; set axial override of infeed axis to 0
; and set 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
;
; Whenever
; current position of oscillation axis equals reversal point 1,
; then
; set axial override of oscillation axis to 0
;and set axial override of infeed axis to 100%
;(this cancels the previous synchronized action!)
WHENEVER $AA_IW[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO
$AA_OVR[Z]=0 $AA_OVR[X]=100
;
; Whenever the distance to go of the partial infeed equals 0,
; then
; set marker with index 0 to 1
; and set marker with index 1 to 1
```

```
 WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1
$AC_MARKER[1]=1
 ;
 ; Whenever         marker with index 0 equals 1,
 ; then
; set axial override of infeed axis to 0, this prevents
; a premature infeed!
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0
;
; Whenever         marker with index 1
; equals 1,
; then
; set axial override of infeed axis to 0,
; (this prevents a premature infeed!)
; and set axial override of oscillation axis to 100%
; (this cancels the previous synchronized action!)
WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
 ;
; If the current position of the oscillation axis is equal to reversal point 1,
; then
; reset marker with index 2,
; enable synchronized action 1
; (no infeed when reaching starting position == reversal point 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 as infeed axis for oscillation axis Z, the X axis should
; infeed to end position 5 in part steps of 1 and the total of all part lengths
; should equal that end position.
   ;
N780 WAITP(Z)
; Release Z axis again
N790 X0 Z0
N799 M30                          ; End of program
```

**Explanations**

The starting position matches reversal point 1. The WHEN .... synchronized actions (in bold above) prevent an infeed when the starting position is reached.

# 6.5 Example of external oscillation reversal

## 6.5.1 Change reversal position via synchronized action with "external oscillation reversal"

```
DEFINE BREAKPZ AS $AA_OSCILL_BREAK_POS1[Z]
DEFINE REVPZ AS $SA_OSCILL_REVERSE_POS1[Z]

WAITP(Z)                    ; Enable oscillation for Z axis
OSP1[Z]=10 OSP2[Z]=60       ; Declare reversal points 1 and 2
OSE[Z]=0                    ; End position = 0
OSB[Z]=0                    ; Starting position = 0

; On external oscillation reversal for oscillation reversal point 1, modify the latter
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 end of oscillation motion
M30
```

■

# Notes

# 7 Data Fields, Lists

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **Axis-specific/spindle-specific** | | **Signals to axis/spindle** | |
| 31, ... | 28.0 | External oscillation reversal | |
| 31, ... | 28.3 | Set reversal point | |
| 31, ... | 28.4 | Alter reversal point | |
| 31, ... | 28.5 | Stop at next reversal point | |
| 31, ... | 28.6 | Stop along braking ramp | |
| 31, ... | 28.7 | PLC controls axis | |
| | | **Signals from axis/spindle** | |
| 31, ... | 100.2 | Oscillation reversal is active | |
| 31, ... | 100.3 | Oscillation cannot start | |
| 31, ... | 100.4 | Error during oscillation movement | |
| 31, ... | 100.5 | Sparking-out active | |
| 31, ... | 100.6 | Oscillation movement active | |
| 31, ... | 100.7 | Oscillation active | |

## 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ... )** | | | |
| 10710 | PROG_SD_RESET_SAVE_TAB | Oscillations to be saved from SD | |
| 11460 | OSCILL_MODE_MASK | Control screen form for asynchronous oscillation | |

## 7.3 Setting data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| **Axisspecific ($SA_ ... )** | | | |
| 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 approached before reversal point 1 after start of oscillation if activated in SD 43770: OSCILL_CTRL_MASK. | |

## 7.4 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, "Diagnostics Guide"
and the online help of MMC 101/102/103 systems.

## 7.5 Main run variables for motion-synchronous actions

The following variables are provided for main run variable_read:

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, axis PCS (Real) |
| **$AA_IB[**<axial expression>**]** | Actual position, axis BCS (Real) |
| **$AA_IM[**<axial expression>**]** | Actual position, axis MCS (IPO setpoints) (Real) With $AA_IM[S1], actual values 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** | Deceleration position after external oscillation reversal when approaching reversal point 1 |
| **$AA_OSCILL_BREAK_POS2** | Deceleration position after external oscillation reversal when approaching reversal point 2 |
| **$AC_TIME** | Time from start of block (Real) in seconds (including times for intermediate blocks generated internally) |
| **$AC_TIMES** | Time from start of block (REAL) in seconds (not including times for intermediate blocks generated internally) |
| **$AC_TIMEC** | Time from start of block (Real) in IPO cycles (including cycles for intermediate blocks generated internally) |
| **$AC_TIMESC** | Time from start of block (Real) in IPO cycles (not including cycles for intermediate blocks generated internally) |
| **$AC_DTBB** | Distance to start of block in BCS (Distance to begin, baseCoor) (Real) |
| **$AC_DTBW** | Distance to start of block in PCS (Distance to begin, workpieceCoor) (Real) |
| **$AA_DTBB[**<axial expression>**]** | Axial distance to start of block in BCS (Distance to begin, baseCoor) (Real) |
| **$AA_DTBW[**<axial expression>**]** | Axial distance to start of block in PCS (Distance to begin, workpieceCoor) (Real) |
| **$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 motion in BCS (Distance to begin, baseCoor) (Real) |
| **$AA_DTEW[**<axial expression>**]** | Axial distance to end of motion in PCS (Distance to end, workpieceCoor) (Real) |
| **$AC_PLTBB** | Path distance to start of block in BCS (Path Length from begin, baseCoor) (Real) |
| **$AC_PLTEB** | Path distance to end of block in BCS (Distance to end) (Path Length to end, baseCoor) (Real) |
| **$AC_VACTB** | Path velocity in BCS (Velocity actual, baseCoor) (Real) |

**$AC_VACTW** Path velocity in PCS
(Velocity actual, workPieceCoor) (Real)
**$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 block start to 1 for block end

**$AA_LOAD[**<axial expression>**]** Drive load (for 611D only)

**$AA_POWER[**<axial expression>**]** Real drive output in W (for 611D only)

**$AA_TORQUE[**<axial expression>**]** Drive torque setpoint in Nm (for 611D only)

**$AA_CURR[**<axial expression>**]** Actual axis current (for 611D only)

**$AC_MARKER[**<arithmetic expression>**]** (int)
Marker variable: Can be used in synchronized actions for creating complex conditions:
There are 8 markers (index 0 – 7).
With reset, the markers are set to 0.
E.g.: WHEN .....DO $AC_MARKER[0]=2
WHEN .....DO $AC_MARKER[0]=3
WHEN $AC_MARKER[0]==3 DO $AC_OVR=50
It is possible to read or write the markers independently of synchronized actions in the parts program:
IF $AC_MARKER == 4 GOTOF SPRUNG

**$AC_PARAM[**<arithmetic expression>**]** (Real)
Floating point parameter for synchronized actions. Used for buffering and evaluation of synchronized actions.
There are 50 parameters (index 0—49) available.

**$AA_OSCILL_REVERSE_POS1[**<axial expression>**]** (Real)
**$AA_OSCILL_REVERSE_POS2[**<axial expression>**]** (Real)
Current reverse positions 1 and 2 for oscillation:
In each case, the current setting data value is read from $SA_OSCILL_REVERSE_POS1 or $SA_OSCILL_REVERSE_POS2.
Changes to the reversal positions in the setting data thus become effective when oscillation is active, i.e. during an active synchronized action.

**Conditions** Conditions for motion-synchronous actions are formulated:

Main run variable Relation operator Expression
For details, please refer to:

**References:** /FB/, S5, “Synchronized Actions”

■

# Notes

# SINUMERIK 840D/840Di/810D Description of Functions Extended Functions (FB2)

# Rotary Axes (R2)



■

## Notes

# 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.

A typical example of a machine tool requiring the use of rotary axes is the 5-axis milling machine. Only with the aid of rotary axes can the tip of the tool be positioned on any point of the workpiece on 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 again after exactly one rotation (modulo 360°). Depending on the application in question, the traversing range of the rotary axis can be limited to less than 360° (e.g. on swivel axes for tool holders) or may be endless (e.g. when tool or workpiece is rotated).

The behavior and features of rotary axes are, in many aspects, identical to those of linear axes. The following functional description is limited to a description of the special features of rotary axes and the differences compared with linear axes.

■

# Notes

# 2 Detailed Description

## 2.1 General

**Definition of rotary axis**

An axis can be declared as a rotary axis by means of machine data IS_ROT_AX. Geometry axes are defined as linear axes. Any attempt to declare them as rotary axes will be rejected with alarm (4200: Geometry axis must not be defined as a 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 simultaneously declared as rotary axes.

**Types of rotary axis**

Depending on the particular application, the operating range of a rotary axis can be endless (i.e. endlessly turning in both directions MD: ROT_IS_MODULO = 1) or restricted by software limit switches (e.g. working area between 0 ... 60°) or limited to a corresponding number of revolutions (e.g. 1000°).

The following list presents some typical applications of rotary axes:

**Typical applications**

- 5-axis machining (operating range limited or unlimited)
- Rotary axis for eccentric machining (unlimited operating range)
- Rotary axis for cylindrical or contour 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; i.e. A rotates around X, B rotates around Y and C rotates around Z (see diagram below). The positive direction of a rotary axis corresponds to a movement to the right looking in the positive axis direction of the corresponding middle axis.

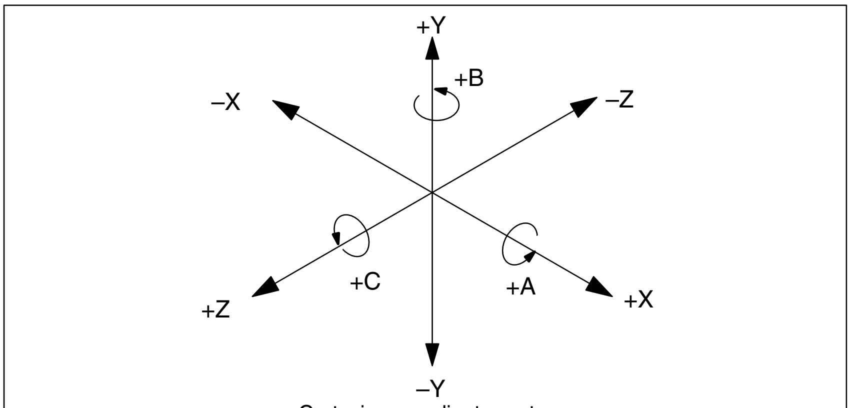



Fig. 2-1 Identification of axes and directions of movements for rotary axes

Extended addressing (e.g. C2=) or freely configured axis addresses can be used for additional rotary axes.

---

**Note**

MD 20050: 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 data inputs and outputs for rotary axes:

Table 2-1 Units of measurement for rotary axes

| Physical quantity | Unit |
|---|---|
| Angular position | Degrees |
| Programmed angular speed | Degrees/minute |
| MD for angular speed | rev/min [1)] |
| MD for angular acceleration | $rev/sec^2$ [1)] |
| MD for angular jerk limitation | $rev/sec^3$ [1)] |

1) These units are interpreted by the control in the axis-specific machine data as soon as the axis is declared as a rotary axis. The user has the option of defining other units for data input/output using machine data.

**References:** /FB/, G2, "Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control"

### Operating range

The axis operating range can be defined by means of axis-specific machine and setting data (software limit switches and working area limitations). As soon as the modulo conversion is activated for the rotary axis (MD: ROT_IS_MODULO = "1"), the operating range is unlimited and the software limit switches and working area limitations are inactive.

In software version 6.3 and higher, the software limit switches/working area limitation can also be activated dynamically by the PLC via interface signal DB31, ... ; DBX 12.4 (initiated by M/H functions in the part program where appropriate). The NC checks back via DB31, ... ; DBX 74.4.

See 2.2

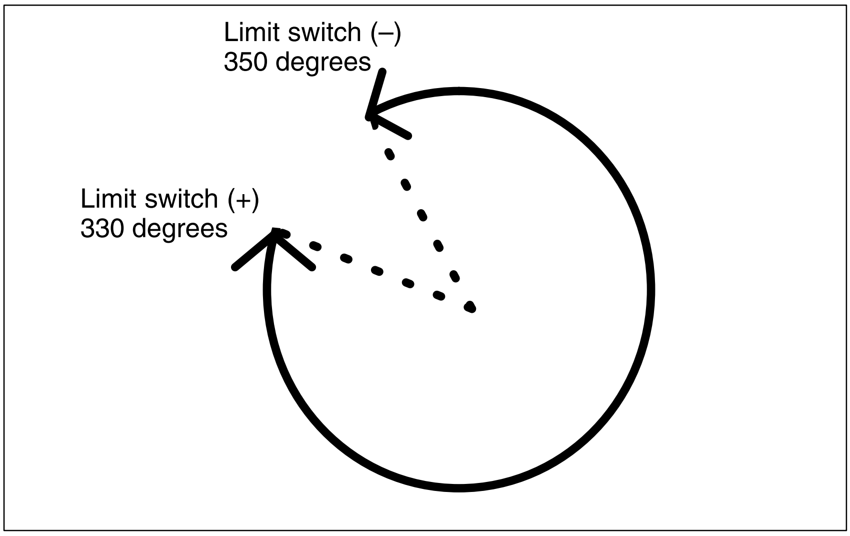



Fig. 2-2 Limited operating area of a modulo rotary axis

**Position display**

The value range for the position display can be set to the modulo 360° representation that is frequently selected for rotary axes (MD: DISPLAY_IS_MODULO = “1”).

**Feed**

The programmed feedrate F corresponds to an angular speed [degrees/min] in the case of rotary axes.

If rotary axes and linear axes traverse along a common path with G94 or G95, the feed should be interpreted in the unit of measurement of the linear axes [e.g. mm/min, inch/min].

The tangential speed of the rotary axis refers to the diameter $D_E$ (unit diameter $D_E=360/\pi$). In the case of unit diameter $D=D_E$, the programmed angular speed in degrees/min and the tangential velocity in mm/min (or inch/min) are numerically identical.

The following applies for the tangential speed in general:

$F = F_W * D / D_E$

F = Tangential speed [mm/min]
$F_W$ = Angular speed [degrees/min]
D = Diameter at which F effective [mm]

where $D_E = 360 / \pi$

$D_E$ = Unit diameter [mm]
$\pi$ = Circle constant Pi

**Revolutional feedrate**

In JOG mode the behavior of the axis/spindle also depends on the setting of setting data JOG_REV_IS_ACTIVE (revolutional feedrate when JOG active).

- If this setting data is active, an axis/spindle is always moved with revolutional feedrate MD JOG_REV_VELO (revolutional feedrate with JOG) or MD JOG_REV_VELO_RAPID (revolutional feedrate with JOG with rapid traverse overlay) depending on the master spindle.
- If the setting data is not active, the behavior of the axis/spindle depends on the setting data ASSIGN_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 JOG_FEED_PER_REV_SOURCE. (In the operating mode JOG, revolutional feedrate for geometry axes on which a frame with rotation is effective).

## 2.2 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 imaging of the axis position internally in the control within the range from 0° to 359.999°. With path settings > 360° (e.g. for incremental dimension programming using G91) the axis position is imaged in the value range between 0° to 360° through a conversion process in the control. The imaging process is applied in JOG and AUTOMATIC mode. The service display is an exception.

In the diagram below, the absolute position of the rotary axis in the positive direction of rotation is represented as a spiral. An arrow marks the actual absolute position (example: Point C‘ = 420°). By sliding the arrow back around the circle (position 0° of the spiral and circle are identical), it is possible to determine a modulo position within the 360° range corresponding to every absolute position. In the example below, absolute position point C’=420° is mapped onto point C = 60° through the modulo conversion process.

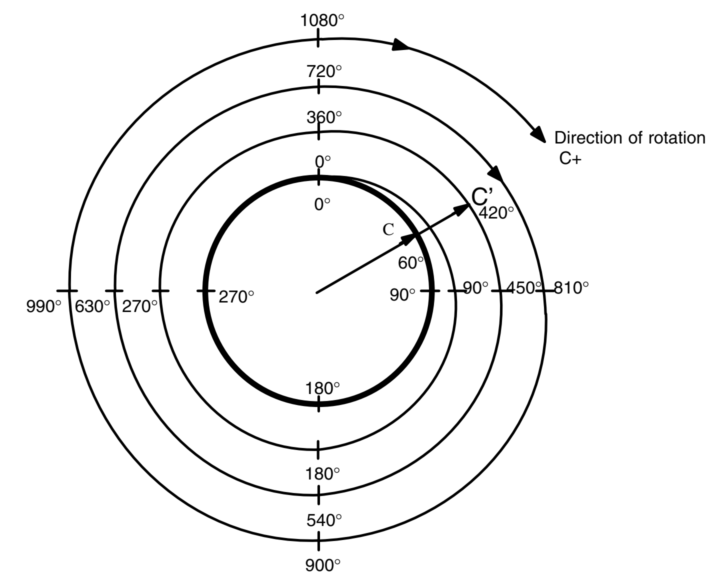



Fig. 2-3 Modulo 360° imaging

**Machine data settings**

Using machine data it is possible to define the programming and positioning settings (MD: ROT_IS_MODULO) and the position display (MD: DISPLAY_IS_MODULO) in modulo 360° for each individual rotary axis to suit the requirements of individual machine tools.

**Axis is modulo**

MD: ROT_IS_MODULO = “1”:

Activation of this machine data allows the special rotary axis action implemented in the system to be utilized (see Subsection 2.3.1), defining The positioning action of the rotary axis for programming (G90, AC, ACP, ACN or DC). A modulo 360° imaging process is executed internally in the control after the current zero offsets have been taken into account. The calculated destination position is subsequently approached **within a single revolution**. The software limit switches and the working area limitations **are ineffective** and the operating range is unlimited (endlessly turning rotary axis).
Please see Section 2.3 on the programming of rotary axes or MD: ROT_IS_MODULO.

**Modulo position display**

MD: DISPLAY_IS_MODULO = 1:
A “modulo 360°”
(1 rotation) position display is frequently required for rotary axes; i.e. when the axis is rotating in the positive direction, the display is periodically reset from° to 0.000° in the control system; with a negative direction of rotation, the axis positions are also displayed in the 0°...359.999° range.

MD: DISPLAY_IS_MODULO = 0:
In contrast to the modulo 360° display method, absolute positions are indicated, e.g. in the positive direction after 1 rotation +360°, after 2 rotations +720° etc. In this case, the display range is limited by the control in accordance with the linear axes.

---

**Note**

The modulo 360° display method should always be selected for a modulo axis (ROT_IS_MODULO = “1”).

---

**Starting position for the modulo rotary axis**

With SW 6.3 and higher, a starting position for the modulo range other than 0 can be defined in MD 30340: MODULO_RANGE_START. It is therefore possible, for example, to define a modulo range of –180° to +180° by entering –180 in MD 30340: MODULO_START_RANGE.

The default setting of 0 (degrees) defines a modulo range of 0–360°.

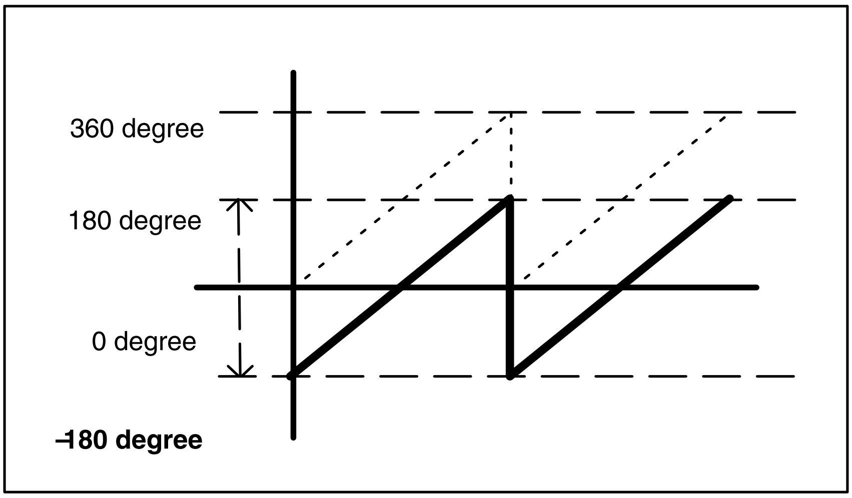



Fig. 2-4 Starting position –180° shifts the modulo range to –180° to + 180°

**Application**

By matching the settings in MD 30503: INDEX_AX_OFFSET and MD 30340: MODULO_RANGE_START, the indexing positions of modulo indexing axes can be implemented analogously to the modulo range.

**References:** /FB/, T1, Indexing Axes

## 2.3 Programming of rotary axes

---

**Note**

For general information on programming, please refer to:
**References:** /PAG/Programming Guide Fundamentals

---

**General**

The machine data ROT_IS_MODULO (modulo conversion for rotary axis) defines whether the rotary axis behaves in the same way as a linear axis during programming and positioning or whether the special features of the rotary axis are incorporated. These features and the differences (mainly with respect to absolute dimension programming) are explained on the following pages.

### 2.3.1 Rotary axis with active modulo conversion (endlessly turning rotary axis)

**Activate modulo conversion**

⇒ MD: ROT_IS_MODULO = “1”
Recommendation: It is also advisable to set the position display to modulo 360º (MD: DISPLAY_IS_MODULO = “1”).

**Absolute dimension programming (AC, ACP, ACN, G90)**

Example of positioning axis: **POS[axis name] = ACP(value)**

- The value identifies the destination position of the rotary axis in a range from 0 to 359.999°. With SW version 6.2 and earlier, alarm 16830 “Incorrect modulo position programmed” is output for values with a negative sign or ≧ 360°.
  With SW version 6.3 and higher, negative values are also possible if the range has been moved with MD 30340: MODULO_RANGE_START and MD 30330: MODULO_RANGE.
- ACP (positive) and ACN (negative) define the traversing direction of the rotary axis unambiguously (irrespective of actual position).
- When programming exclusively with AC or with G90, the traversing direction depends on the actual position of the rotary axis. If the destination position is larger than the actual position, the axis traverses in the positive direction, otherwise it traverses in the negative direction.
  With SW 6.3 and higher, the positioning behavior can be configured in MD 30455: MISC_FUNCTION_MASK Bit 2.
  Bit 2 = 0: Modulo axis positioned per default by AC with G90
  Bit 2 = 1: Modulo axis positioned per default by DC with G90
  (shortest path)
- Use of ACP and ACN: With asymmetrical workpieces, it must be possible to define the traversing direction to prevent collisions during rotation.

**Example:**
(see diagram below): Start position of C is 0˚

| | Programming | Effect |
|---|---|---|
| **1** | POS[C] = ACP(100) | Rotary axis C traverses in the positive rotational direction to position 100˚ |
| **2** | POS[C] = ACN(300) | C traverses in the negative rotational direction to position 300˚ |
| **3** | POS[C] = ACP(240) | C traverses in the positive rotational direction to position 240˚ |
| **4** | POS[C] = AC (0) | C traverses in the negative rotational direction to position 0˚ |

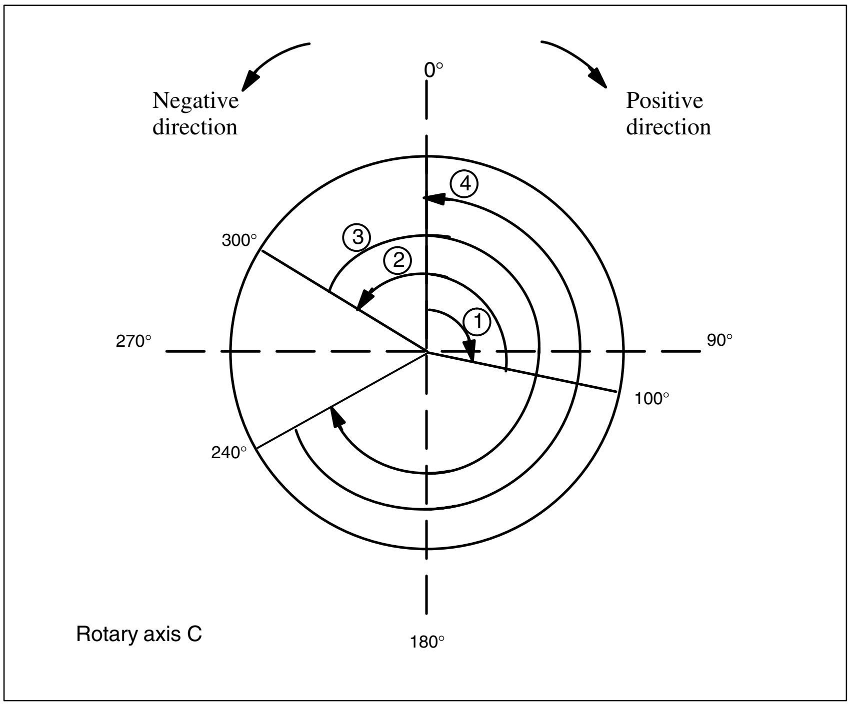



Fig. 2-5 Examples of absolute dimension programming for modulo axes

**Absolute dimension programming via the shortest route (DC)**

**POS[axis name] = DC(value)**

– The value identifies the destination position of the rotary axis 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 via the **shortest route** within one revolution (traversing movement max. ±180˚).

– The control calculates the direction of rotation and the traversing distance according to the actual position. If the distance to be traversed is the same in both directions (180˚), the positive direction receives preference.

– Example application of DC: the rotary table is required to approach the changeover position in the shortest time (and therefore via the shortest path).

– If DC is programmed with a linear axis, alarm 16800 “DC traversing instruction cannot be used” is output.

**Example:**
(see diagram below): Start position of C is 0˚

| | **Programming** | **Effect** |
|---|---|---|
| **1** | POS[C] = DC(100) | C axis traverses along the shortest path to position 100˚ |
| **2** | POS[C] = DC(300) | C axis traverses along the shortest path to position 300˚ |
| **3** | POS[C] = DC(240) | C axis traverses along the shortest path to position 240˚ |
| **4** | POS[C] = DC (60) | C axis traverses along the shortest path to position 60˚. Since the distance in this case is equal to 180˚ in both directions, the positive direction is the preferred direction of rotation. |

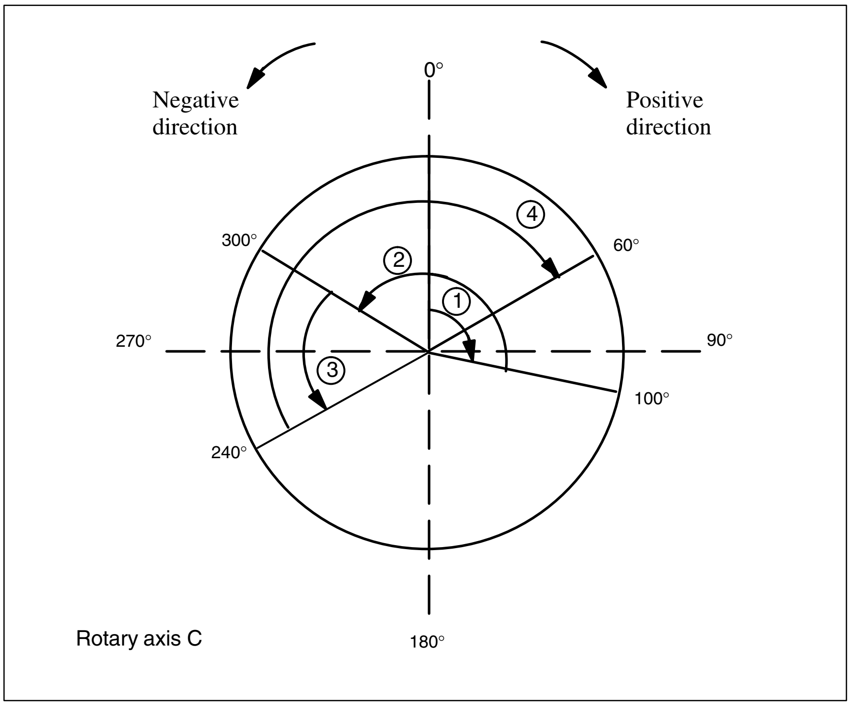



Fig. 2-6 Examples of DC programming

### Block search response

After a block search with calculation, the accumulated search position of the modulo conversion can be interrogated via system variable $AC_RETPOINT. The position returned by the system variable is **not** modulo-converted and the programmed number of revolutions would be lost.

**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**

In **software version 6.3** and higher, the working area limitation/software limit switches can be activated/deactivated dynamically by the PLC for a modulo rotary axis via interface signal DB31, ... ; DBX 12.4 (as in the case of rotary axes). The NC feeds back the current status of the travel limitation via DB31, ... ; DBX 74.4.

The monitoring function is activated if interface signal DB31, ... ; DBX 12.4 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 switchover are monitored.

**Supplementary conditions:**

The software limit switch monitor can be activated or deactivated via the PLC interface for modulo axes only.

Travel 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 activated/deactivated in pairs. To guarantee correct monitoring of the working area limitations, **both** limitations must have been activated either via G26/G25 or SD 43400: WORKAREA_PLUS_ENABLE and SD 43410: WORKAREA_MINUS_ENABLE.

**Example of a travel limitation switchover**

Two workholders must be machined consecutively on a modulo rotary axis. The first workholder has a number of clamped workpieces. It is then replaced by a workholder with a built-on axis whose working area must be monitored to prevent damage to supply lines.

Configuration:

$MA_IS_ROT_AX[AX4] = 1
$MA_ROT_IS_MODULO[AX4] = 1
$MA_POS_LIMIT_PLUS[AX4] = 340
$MA_POS_LIMIT_MINUS[AX4] = 350

Extract of parts program:

```
M123                ;Mechanically insert workholder with four clampings
                    ;Deactivate software limit switches in the B axis from
                    ;the PLC DB35, DBX12.4=0
STOPRE                              ;Initiate 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                ;Mechanically insert workholder with built-on axis
                    ;Activate software limit switches in the B axis from
                    ;the PLC DB35, DBX12.4=1
STOPRE              ;Initiate a preprocessing stop
B270
```

### Incremental dimension programming (IC, G91)

Example of positioning axis: **POS[axis name] = IC(+/-value)**

– The value identifies the traversing distance of the rotary axis. The value can be negative as well as ≥ +/–360°.

– The **leading sign** of the value defines the**traversing direction** of the rotary axis.

– Sample application: Milling a spiral groove across several revolutions

**Example:**

| | |
|---|---|
| POS[C] = IC(720) | C axis traverses incrementally in positive direction through 720° (2 revolutions) |
| POS[C] = IC(–180) | C axis traverses incrementally in negative direction through 180° |

### 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:**

```
LOOP:
POS[C] = IC(720)
GOTOB LOOP
```

## 2.3.2 Rotary axis without modulo conversion

**Deactivate modulo conversion**

⇒ MD: Set ROT_IS_MODULO = “0”

**Absolute dimension programming (AC, G90)**

Example of positioning axis: **POS[axis name] = AC(+/–value)**

– The value and its leading sign provide a unique identification of the destination position of the rotary axis. The value can also be ≥ +/–360°. The position value is limited by the software limit switch positions.

– The traversing direction is ascertained by the control according to the leading sign of the actual position of the rotary axis.

– If ACP or ACN are programmed, alarm 16810 “ACP traversing instruction cannot be used” or alarm 16820 “ACN traversing instruction cannot be used” is output.

– Sample application: Linear movements are incorporated in the rotary axis (cam gear); certain end positions may therefore not be overtraveled.

**Example**:

| Programming | Effect |
|---|---|
| POS[C] = AC (–100) | Rotary axis C approaches position –100°; the traversing direction depends on the starting position |
| POS[C] = AC (1500) | Rotary axis C traverses to the position at 1500°; |

**Absolute dimension programming across 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 destination position of the rotary axis **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 ≥ 360°.

– With DC (Direct Control), the rotary axis approaches the programmed absolute position via the **shortest route** within one revolution (traversing movement max. ±180°).

– The control calculates the direction of rotation and the traversing distance according to the actual position (in relation to modulo 360°). If the distance to be traversed is the same in both directions (180°), the positive direction receives preference.

– Example application of DC: the rotary table is required to approach the changeover position in the shortest time (and therefore via the shortest route).

– If DC is programmed with a linear axis, alarm 16800 “DC traversing instruction cannot be used” is output.

**Example:**

| Programming | Effect |
|---|---|
| POS[C] = AC (7200) | Rotary axis C traverses to position 7200˚; the traversing direction depends on the starting position |
| POS[C] = DC (300) | Rotary axis C traverses via the shortest route to "modulo" position 300˚. C therefore traverses through 60˚ in the negative direction and stops at the absolute position 7140˚. |
| POS[C] = AC (7000) | Rotary axis C traverses to the absolute position 7000˚; here, C traverses through 140˚ in the negative direction of rotation |

Note: In this example, it is advisable to activate the modulo 360º display (MD: DISPLAY_IS_MODULO = "1").

**Incremental dimension programming (IC, G91)**

Example of 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 traversing distance of the rotary axis. The value can be negative as well as ≥ +/–360˚.

– The **leading sign** of the value defines **the direction of travel** of the rotary axis.
For example, see Subsection 2.3.2.

**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.

## 2.3.3 Miscellaneous 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 system always bases its modulo calculation on a 360º full circle.

**Preset actual value memory**

PRESETON is possible

**Indexing axes**

**References:** /FB/, T1, "Indexing Axes"

## 2.4 Start-up of rotary axes

**Procedure**

The procedure for starting up rotary axes is identical to that for linear axes with a small number of exceptions. It should be noted that the units of the axis-specific machine and setting data on the control are interpreted as follows as soon as the axis has been defined as a rotary axis (MD: IS_ROT_AX = 1):

| | |
|---|---|
| Position | in “degrees” |
| Velocity | in “rev/minute” |
| Acceleration | in “rev/seconds$^2$” |
| Jerk limitation | in “rev/seconds$^3$” |

**Special MD**

The special machine data of the rotary axis described in Chapter 4 must also be entered depending on the application.

- MD: ROT_IS_MODULO — Modulo conversion for positioning and and programming
- MD: DISPLAY_IS_MODULO — Modulo conversion for position display
- MD: INT_INCR_PER_DEG — Precision of angular position calculation

The following overview lists the possible combinations of these machine data for a rotary axis.

Table 2-2 Possibilities for combining machine data of rotary axes

| MD: IS_ROT_AX “rotary axis” | MD: ROT_IS_MODULO “Modulo conversion for rotary axis” | MD: DISPLAY_IS _MODULO “Modulo actual value display” | Application permitted | Remarks |
|---|---|---|---|---|
| 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: e.g. for axes with an operating range of +/–1000° |
| 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**). With SW 6.3 and later, the axis can be utilized with or without working area limitation. See 2.3.1. |

Table 2-2 Possibilities for combining machine data of rotary axes

| **MD: IS_ROT_AX "rotary axis"** | **MD: ROT_IS_MODULO "Modulo conversion for rotary axis"** | **MD: DISPLAY_IS _MODULO "Modulo actual value display"** | **Application permitted** | **Remarks** |
|---|---|---|---|---|
| 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.<br>With SW 6.3 and later, the axis can be utilized with or without working area limitation. See 2.3.1. |
| 0 | 0 or 1 | 0 or 1 | not recommended | Axis is not a rotary axis; the other MD are not therefore evaluated. |

### JOG velocity for rotary axes

With SD: JOG_ROT_AX_SET_VELO (JOG velocity for rotary axes), a jog velocity that is valid for all rotary axes can be set.

If a value of 0 is entered in the setting data, then axial MD: JOG_VELO (JOG axis velocity).

**References:** /FB/, H1, "Manual and Handwheel Travel"

## 2.5 Special features of rotary axes

**Software limit switches**

The software limit switches and working area limitations are operative and are required for swivel axes with a restricted operating range. For endlessly turning rotary axes with (MD: ROT_IS_MODULO=1), however, the software limit switches and working area limitations are set inactive.

With SW 6.3 and later, a modulo rotary axis can be utilized with or without working area limitation. See 2.3.1. Dynamic switchover by the PLC.

**References:** /FB/, A3, “Axis Monitoring”

**Mirroring of rotary axes**

Mirroring can be implemented for rotary axes with programming commands MIRROR(C) and AMIRROR(C).

**Reference point approach**

**References:** /FB/, R1, “Reference Point Approach”

**Spindles as rotary axes**

For notes concerning the use of spindles and rotary axes (C axis operation), please refer to:

**References:** /FB/, S1, “Spindles”

■

# Notes

# 3 Supplementary Conditions

There are no supplementary conditions stipulated for this Description of Functions.

■

# 4 Data Descriptions (MD, SD)

## 4.1 Axis/spindlespecific machine data

| | | | |
|---|---|---|---|
| **30300**<br>MD number | **IS_ROT_AX**<br>Rotary axis | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after POWER ON | Protection level: 2 | Unit: – | |
| Data type: BOOLEAN | Applies from SW: 1.1 | | |
| Meaning: | 1: Axis: The axis is defined as a "rotary axis".<br>● The special functions of the rotary axis are active or can be activated by means of additional machine data according to the type of machine required (see below.<br>● The unit of measurement is degrees.<br>● The units of the axis-specific machine and setting data are interpreted as follows by the control system when the default setting is applied:<br>● Position in degrees<br>● Velocity in rev/minute<br>● Acceleration in rev/$s^2$<br>● Jerk limitation in rev/$s^3$<br>Spindle:<br>The machine data must always be set to "1" for a spindle, otherwise alarm 4210 "rotary axis declaration missing" is output.<br>0: The axis is defined as a "linear axis". | | |
| Special cases, errors, ...... | For axis: alarm 4200 if the axis is already defined as a geometry axis.<br>For spindle: Alarm 4210 | | |
| Related to .... | The following machine data are effective only after activation of MD: IS_ROT_AX = "1":<br>● MD: ROT_IS_MODULO "Modulo conversion for rotary axis"<br>● MD: DISPLAY_IS_MODULO "Position display is modulo"<br>● MD: INT_INCR_PER_DEG "Calculation precision for angle positions" | | |
| References | Tab. 2.2 Combination options for machine data | | |

| 30310<br>MD number | ROT_IS_MODULO<br>Modulo conversion for rotary axis | | | |
|---|---|---|---|---|
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 |
| Changes effective after POWER ON | | | Protection level: 2 | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 1.1 | |
| Meaning: | 1: A modulo conversion is performed on the setpoints for the rotary axis.<br>The software limit switches and the working area limitations are **inoperative**; the traversing range is therefore unlimited in both directions.<br>With SW 6.3 and later, activation of the working area limitations/software limit switches by the PLC can be enabled/disabled dynamically.<br>MD: IS_ROT_AX must be set to “1”<br><br>For further information, see Section 2.2<br><br>0: No modulo conversion | | | |
| MD irrelevant for ...... | MD: IS_ROT_AX = “0” (linear axes) | | | |
| Tab. 2.2 | Combination options for machine data | | | |
| Application example(s) | Continuously rotating axes (e.g. for eccentric rotation, grinding, winding) | | | |
| Related to .... | MD: DISPLAY_IS_MODULO “Position display is modulo 360°”<br>MD: IS_ROT_AX = 1 “Rotary axis”<br>MD: POS_LIMIT_MINUS “Software limit switch minus”<br>MD: POS_LIMIT_PLUS “Software limit switch plus”<br>SD: WORKAREA_LIMIT_MINUS “Working area limitation minus”<br>SD: WORKAREA_LIMIT_PLUS “Working area limitation plus” | | | |

| 30320<br>MD number | DISPLAY_IS_MODULO<br>Position display is modulo 360° | | | |
|---|---|---|---|---|
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 1 |
| Changes effective after POWER ON | | | Protection level: 2 | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 1.1 | |
| Meaning: | 1: Position display “Modulo 360º” is active:<br>The position display of the rotary axis or spindle (for basic or machine coordinate system) is defined as “Modulo 360°”. In the positive direction of rotation, therefore, the control resets the position display internally<br>to 0.000 degrees following each cycle of 359.999<br>degrees. The display range is always positive and always between 0° and 359.999°.<br>With **SW 6.3** and higher, the modulo range can be shifted in MD 30340: MODULO_RANGE_START.<br>Its limits are then defined by the settings in MD 30340: MODULO_RANGE_START and MD 30330: MODULO_RANGE.<br>0: Absolute position display is active:<br>In contrast to to the modulo 360° method, the absolute position display shows, e.g. in a positive direction, +360° after 1 rotation,<br>+720° after 2 rotations, etc.<br>In this case, the display range is limited in line with the linear axes. | | | |
| MD irrelevant for ...... | Linear axes MD: IS_ROT_AX = “0” | | | |
| Application example(s) | • With continuously rotating axes (MD: ROT_IS_MODULO = “1”), it is also advisable to activate the position display with modulo 360° .<br>• The position display for spindles must always be activated with modulo 360°. | | | |
| Related to .... | MD: IS_ROT_AX = 1 “axis is rotary axis”<br>MD 30340: MODULO_RANGE_START<br>MD 30330: MODULO_RANGE | | | |

| | | | | | |
|---|---|---|---|---|---|
| **30330**<br>MD number | **MODULO_RANGE**<br>Size of modulo range | | | | |
| Default setting: 360.0 | | Minimum input limit: 1.0 | | Maximum input limit: 360000000.0 | |
| Changes effective after RESET | | | Protection level: 2 / 7 | | Unit: Degrees |
| Data type: DOUBLE | | | | Applies from SW: SW 4.1 | |
| Meaning: | The MD defines the size of the modulo range. Position inputs are accepted and displayed within this range. Meaningful modulo range values are n * 360 degrees. Other settings are also possible in principle, but it must be ensured that there is a meaningful relation between the positions in the NC and the mechanical setup (ambiguity).<br>Velocity specifications are not affected by the settings in this MD. | | | | |
| Related to .... | MD 30340: MODULO_RANGE_START (SW 6.3) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **30340**<br>MD number | **MODULO_RANGE_START**<br>Modulo range starting position | | | | |
| Default setting: 0 | | Minimum input limit: Minus | | Maximum input limit: Plus | |
| Changes effective after RESET | | | Protection level: 2 / 7 | | Unit: Degrees |
| Data type: DOUBLE | | | | Applies from SW: 6.3 | |
| Meaning: | A starting position can be specified for the module range<br>$MA_MODULO_RANGE of the rotary axes.<br><br>Example:<br>Modulo range between 0 and 360 degrees (default)<br>MODULO_RANGE = 360<br>MODULO_RANGE_START = 0<br><br>Modulo range between –180 degrees and +180 degrees<br>MODULO_RANGE = 360<br>MODULO_RANGE_START = –180<br><br>These two machine data are also used to calculate the reference point position for rotary axes with rotary, distance-coded encoders. The reference point position is adapted to the travel limits of the modulo range when<br>MD 30455: MISC_FUNCTION_MASK Bit 1 = 1. | | | | |
| Related to .... | MD 30330: MODULO_RANGE<br>MD 30455: MISC_FUNCTION_MASK | | | | |

| **30455**<br>MD number | | **MISC_FUNCTION_MASK**<br>Axis functions | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: 0x00 | | Maximum input limit: 0x10 |
| Changes effective after RESET | | Protection level: 2 / 7 | Unit: – |
| Data type: DWORD | | Applies from SW: 6.3 (Bit 4 expanded)<br>SW version 6.4 and higher (bit 3 expanded) | |

Meaning:

This machine data specifies the following axis functions in more detail:

| | | |
|---|---|---|
| Bit 0 | Modulo rotary axis programming | Modulo rotary axis/spindle |
| Bit 1 | Reference point definition | Rotary, distance-coded encoders |
| Bit 2 | Modulo rotary axis positioning | Modulo rotary axis/spindle |
| Bit 3 | Setpoint or actual value axis positions | With spindle / axis disable (SW 6.4 and higher) |
| Bit 4 | Feed enable with synchronous spindle | Following spindle (SW 6.3 and higher) |

For rotary axes with:
MD 30310: ROT_IS_MODULO = 0 which utilize rotary, distance-coded encoders,
MD 34210: ENC_REFP_MODE = 3, the reference point position is calculated as a funct. of
MD 30330: MODULO_RANGE and MD 30340: MODULO_RANGE_START.
This is automatically adapted to the travel limits of the modulo range.

For rotary axes with:
MD 30310: ROT_IS_MODULO = 1, the bit has no meaning since the reference point position is always adapted within the modulo range.

| Bit | | **Effect:** |
|---|---|---|
| 1 | 0 | |
| | | Modulo rotary axis/spindle: |
| 0 | 0 | Programmed positions must be within the modulo range. An alarm is otherwise generated. |
| 0 | 1 | No alarm is generated if positions are programmed outside the modulo range. The modulo conversion is performed internally. |

**Example:**

| | |
|---|---|
| B–5 has | the same meaning as B355, |
| POS[A] = 730 | is identical to POS[A] = 10 and |
| SPOS = –360 | behaves like SPOS = 0 (modulo range 360 degrees) |

| | | |
|---|---|---|
| | | Reference point definition: |
| 1 | 0 | Definition of reference point position of rotary, distance-coded encoders analogous (1:1) to mechanical absolute position. |
| 1 | 1 | Definition of reference point position of rotary, distance-coded encoders within the configured modulo range. |
| | | Modulo rotary axis/spindle: |
| 2 | 0 | Modulo rotary axis positioned per default by AC with G90 |
| 2 | 1 | Modulo rotary axis positioned per default by DC with G90 (shortest pos.) |
| 3 | 0 | With spindle / axis disable, $VA_IM, $VA_IM1, $VA_IM2 returns the setpoint. |
| 3 | 1 | With spindle / axis disable, $VA_IM, $VA_IM1, $VA_IM2 returns the actual value. |
| | | Feed enable with synchronous spindle |
| 4 | 0 | Synchronous spindle coupling, following spindle: Cancelation of feed enable brakes the coupled grouping. |
| 4 | 1 | Following spindle: Feed enable refers only to the interpolation component of the overlaid movement (SPOS),..) and does not affect the coupling. |

Related to ....

MD 30310: ROT_IS_MODULO
MD 30330: MODULO_RANGE
MD 30440: MODULO_RANGE_START

| | |
|---|---|
| **34220**<br>MD number | **ENC_ABS_TURNS_MODULO[n]**<br>Absolute encoder range for rotary encoders: 0 ... max. no. of encoders –1 |
| Default setting: 4096, 4096 | Minimum input limit: 1 / Maximum input limit: 4096 |
| Changes effective after POWER ON | Protection level: 2 / 7 / Unit: – |
| Data type: DWORD | Applies from SW: 2.2 |
| Meaning: | The absolute position of a rotary axis is reduced to the following range after an absolute encoder is switched on:<br>i.e. a MODULO conversion is performed if the read actual position is greater than the position allowed by the setting in MD ENC_ABS_TURNS_MOTOR.<br><br>0 degrees <= position <= n*360 degrees (where n = ENC_ABS_TURNS_MODULO)<br>**Note:** In SW 2.2 the position is reduced to this range when the control/encoder is switched on. With SW 3.6 and higher, half this value represents the maximum permissible traversing path when the control or encoder is switched off. |
| Special cases, errors, ...... | Only powers of two are allowed as values ( 1, 2, 4, 8, 16, ..., 4096).<br>If other values are entered, they are “rounded off” without< a message in SW version 4.1 and earlier. In SW 4.1 and higher, rounding off is shown in the machine data; the change is displayed in alarm 26025.<br><br>**The MD is only relevant for rotary encoders (with linear and rotary axes).**<br><br>**Important recommendation:**<br>The default value “1 encoder revolution” was changed to “4096” from software version 3.6. The new value provides a more robust setting for the most commonly used encoder types. When using an encoder with less multi-turn information, or when using single-turn encoders, the value must be decreased accordingly. In any case, for multi-turn absolute encoders the value should be changed to the maximum quantity supported by the encoder so that the unambiguous traversing range that is increased as a result can be utilized (Note: This value also influences the permissible position offset when the encoder or power supply is switched off). |
| Related to .... | Drive MD 1021, ENC_ABS_TURNS_MOTOR,<br>Drive MD 1031, ENC_ABS_TURNS_DIRECT |

■

# Notes

# 5 Signal Descriptions

| DB 31, ... ; **DBX12.4** Data Block | Traversing range limitation for modulo rotary axes<br>Signal(s) from PLC to NCK axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.3 |
| Signal state 1 or signal transition 0 ——> 1 | Activate travel limitation for modulo rotary axis (software limit switches, working area limitations) | |
| Signal state 0 or signal transition 1 ——> 0 | Deactivate travel limitation for modulo rotary axis | |
| Signal irrelevant for ... ... | Linear axes / rotary axes without modulo functionality | |
| Application example(s) | Built-on rotary axis with monitoring | |

| DB 31, ... ; **DBX74.4** Data Block | Monitoring status with modulo rotary axes<br>Signal(s) from NCK to PLC axis/spindle | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.3 |
| Signal state 1 or signal transition 0 ——> 1 | Travel limitation for modulo rotary axis active (software limit switches, working area limitations) | |
| Signal state 0 or signal transition 1 ——> 0 | Travel limitation for modulo rotary axis not active | |
| Signal irrelevant for ... ... | Linear axes / rotary axes without modulo functionality | |
| Application example(s) | Built-on rotary axis with monitoring | |

■

# 6 Example

**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 of the workpiece and take up any position on the tool axis. Various milling heads are required according to the application. Fig. 6-1 illustrates a fork head and an inclined axis head as example arrangements for rotary axes.

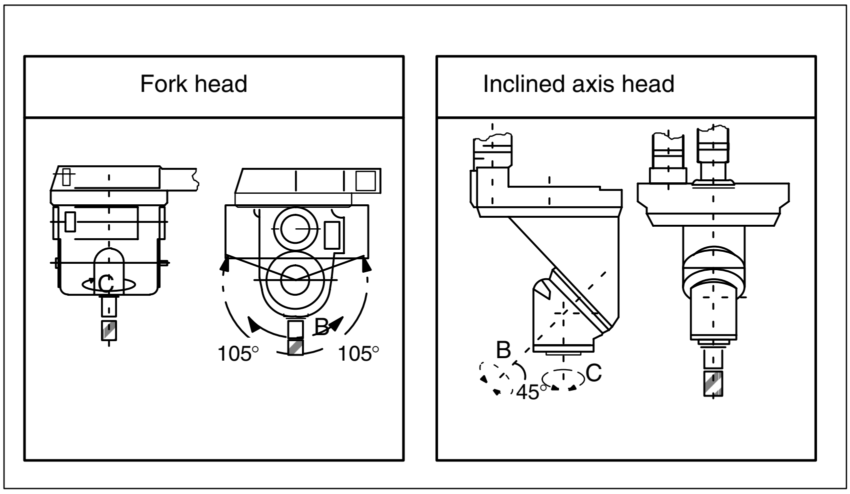



Fig. 6-1 Fork head, inclined axis head

■

# Data Fields, Lists 7

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **Axis-specific** | | | |
| 31, ... | 12.4 | Traversing range limitation for modulo axis | |
| 31, ... | 74.4 | Status of SW limit switch monitoring for modulo axis | |

## 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ... )** | | | |
| 10210 | INT_INCR_PER_DEG | Calculation resolution for angular positions | G2 |
| **Axis/channelspecific ($MA_ ... )** | | | |
| 30320 | DISPLAY_IS_MODULO | Actual-value display modulo | |
| 30300 | IS_ROT_AX | Axis is rotary axis | |
| 36100 | POS_LIMIT_MINUS | Software limit switch minus | A3 |
| 36110 | POS_LIMIT_PLUS | Software limit switch plus | A3 |
| 30310 | ROT_IS_MODULO | Modulo conversion for rotary axis | |
| 30330 | MODULO_RANGE | Size of the modulo range | |
| 30340 | MODULO_RANGE_START | Starting position for the modulo range | |
| 30455 | MISC_FUNCTION_MASK | Axis functions | |

## 7.3 Setting data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($SN_ ...)** | | | |
| 41130 | JOG_ROT_AX_SET_VELO | JOG speed for rotary axes | H1 |

| Number | Names | Name | Reference |
|---|---|---|---|
| **Axis-specific ($SA_ ... )** | | | |
| 43430 | WORKAREA_LIMIT_MINUS | Working area limitation minus | A3 |
| 43420 | WORKAREA_LIMIT_PLUS | Working area limitation plus | A3 |

## 7.4 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, "Diagnostics Guide"
and the online help of MMC 101/102/103 systems.

■

# SINUMERIK 840D/840Di/810D
# Description of Functions
# Extended Functions (FB2)

# Synchronous Spindle (S3)

■

# 1 Brief Description

**Synchronous spindle**

This function (see Chapter 3) enables synchronization between a leading and following spindle with high angle accuracy.

It also offers the option of on-the-fly transfer of the workpiece from spindle 1 to spindle 2 during operation on turning machines for the purpose, for example, of final machining. Advantage: Avoidance of downtimes.

Apart from synchronizing the spindle speed, it is also possible to specify the relative angular position of the spindles in relation to one another, e.g. for on-the-fly, position-oriented transfer of workpieces.

On-the-fly transfer of workpieces between leading spindle (LS) and following spindle (FS):

- $n_{FS} = n_{LS}$ Speed synchronism
- $\varphi_{FS} = \varphi_{LS}$ Position synchronism

Or

- $\varphi_{FS} = \varphi_{LS} + \Delta\varphi$ Position synchronism with angular offset

**Polygonal machining**

As an additional feature, specification of an integer, multiple speed ratio $k_Ü$ between the main spindle and a "tool spindle" provides the basis for polygonal machining (polygonal turning).

Polygonal turning:

- $n_{FS} = k_Ü \cdot n_{LS}$

Synchronous operation is selected and deselected via the CNC parts program.

The synchronous spindle pairs for each machine can be assigned a fixed configuration by means of channel-specific machine data or defined for specific applications via the CNC parts program.

Up to two synchronous spindle pairs can be operated in each NC channel.

**SW 5 and higher**

Any number of following spindles in any channels on an NCU can be coupled to one leading spindle. The only possible restriction could be imposed by the real CPU time requirement.

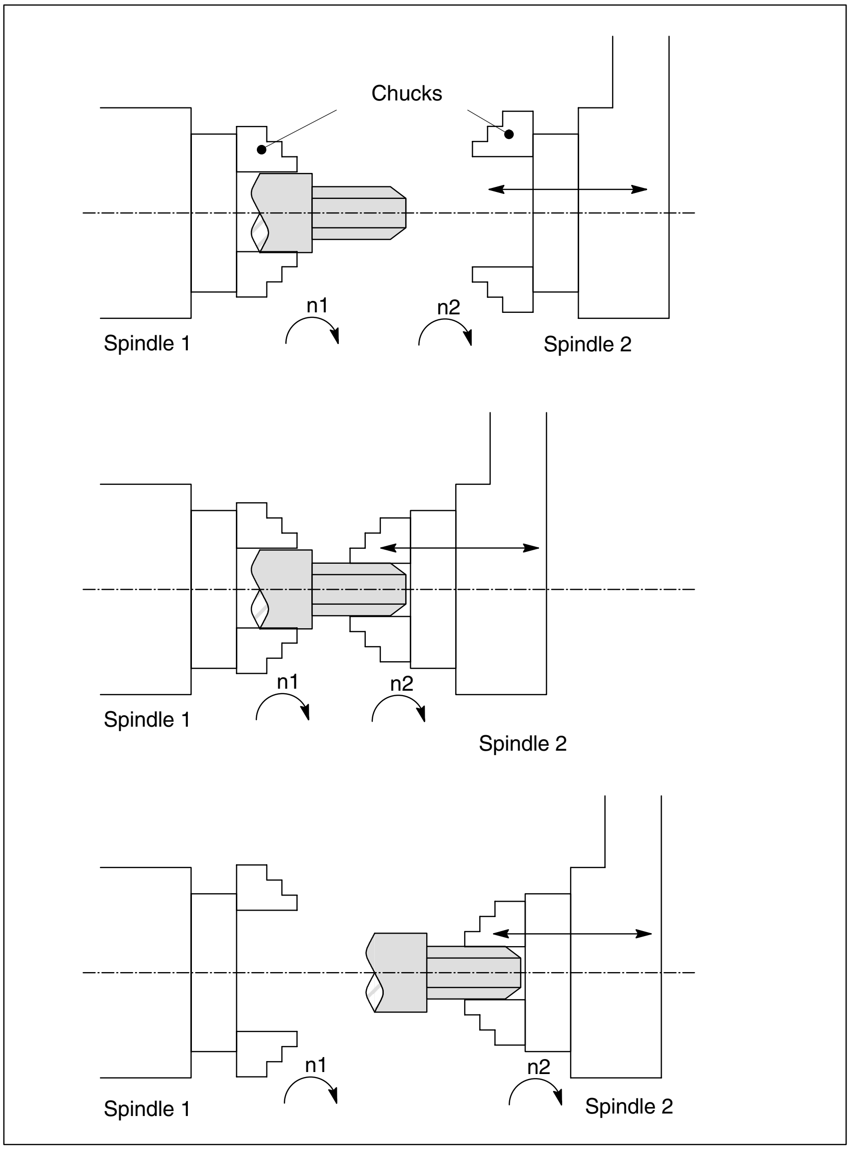



Fig. 1-1 Synchronous mode, on-the-fly workpiece transfer from spindle 1 to spindle 2

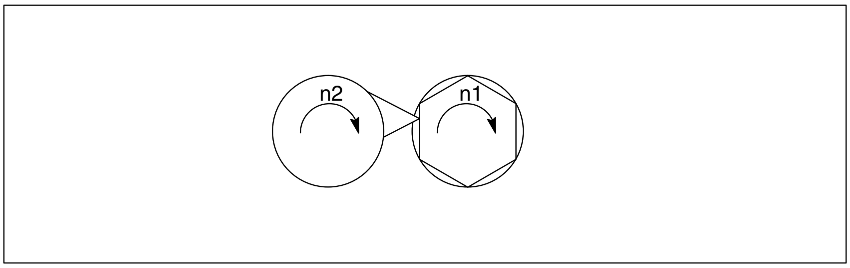



Fig. 1-2 Polygonal turning

■

# Detailed Description 2

## 2.1 General functionality

### 2.1.1 Synchronous mode

**Explanations**

| | |
|---|---|
| <Axial expression>: | can be:<br>– Axis identifier<br>– Spindle identifier |
| <Axis identifier>: | C (if spindle 1 has the identifier "C" in axis operation.) |
| <Spindle identifier>: | Sn, SPI(n) mit n = Spindle number |
| <Spindle number>: | 1, 2, ... according to the spindle number defined in<br>MD 35000: SPIND_ASSIGN_TO_MACHAX |
| (FS, LS, Offset): | LS = Leading Spindle, FS = Following Spindle,<br>Offset = read programmable offset of following spindle using system variables |
| $P_COUP_OFFS[Sn] | Programmed position offset of 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 to back to open-loop control mode.

As soon as synchronous mode is active for the following spindle,
IS "Synchronous mode" (DB31, ... DBX84.4) = 1 is signaled to the PLC.

**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 versions of the appropriate software versions.

- **In SW 4 and lower**, **two** synchronous spindle couplings can be operated in each NC channel.
- **In SW 5 and higher**, any number of following spindles in any channels of one NCU can be coupled to this leading spindle.
- **In SW 6 and higher**, any number of following spindles in any channels of one NCU or a different NCU can be coupled to this leading spindle.

**Options in synchronous mode**

The following functions are available for synchronous mode:

- Following and leading spindle rotate at the same speed ($n_{FS} = n_{LS}$ ; speed ratio $k_{Ü} = 1$)
- LS and FS rotating in the same direction or in opposite directions (can be defined by specifying positive or negative speed ratio $k_{Ü}$)
- Following and leading spindles rotate at different speeds ($n_{FS} = k_{Ü} \cdot n_{LS}$ ; speed ratio $k_{Ü} \neq 1$)
  Application: Polygonal turning
- Adjustable angular position between FS and LS ($\varphi_{FS} = \varphi_{LS} + \Delta\varphi$ )
  The spindles rotate at synchronous speed with a defined angular offset between the LS and FS (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 speed 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 both

- a fixed configuration in channel-specific machine data (referred to below as **“fixed coupling configuration”**) and
- a freely defined coupling via language instructions (COUP...) in the parts program (referred to below 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 parts program.
2. No coupling is configured via machine data. In this case, the couplings can be user-defined and parameterized via the parts program.

**SW 6 and higher**

The special **following spindle interpolator** allows a number of following spindles from different channels or another NCU to be coupled as defined by the user to a single leading spindle. The following spindle interpolator is

- activated by COUPON and
- deactivated by COUPOF

and is always located in the channel in which the COUPON statement has been programmed for the following spindle. If the following spindle to be activated was previously programmed in another channel, COUPON 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. The latter act exclusively on the slave spindles and do not affect the leading spindle. For further information, please see Subsection 2.1.5.

**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. Fixed coupling configuration:

   Machine axes which are to function as the following spindle (FS) and leading spindle (LS) are defined in channel-specific MD 21300: COUPLE_AXIS_1[n].

   The machine axes programmed as the LS and FS for this coupling configuration cannot be altered by the NC parts program.

   If necessary, the coupling parameters can be modified with the NC parts program.

2. User-defined coupling:

   Couplings can be created and altered in the NC parts program with language instruction "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 instruction COUPDEL(FS,LS)).

   The axis identifiers (Sn, SPI(n)) for the following and leading spindles must be programmed with FS and LS for every language instruction COUP**...**, thus ensuring that the synchronous spindle coupling is unambiguously defined.

   The valid spindle number must be assigned to a machine axis in axis-specific MD 35000: 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.

**Speed ratio**

The speed ratio is programmed with separate numerical values for numerator and denominator (speed ratio parameters). It is therefore possible to specify the speed ratio very exactly, even with rational numbers.

In general: $k_{Ü} = \dfrac{\text{speed ratio parameter numerator}}{\text{speed ratio parameter denominator}} = \dfrac{Ü_{numerator}}{Ü_{denominator}}$

The value range of the speed ratio parameter ($Ü_{numerator}$, $Ü_{denominator}$) is virtually unlimited internally in the control.

The speed ratio parameters for the coupling configured via machine data can be defined in channel-specific SD 42300: COUPLE_RATIO_1[n]. In addition, the ratio can be altered with language instruction COUPDEF(FS, LS, $Ü_{numerator}$, $Ü_{denominator}$ ,**...**). The values entered in the setting data are not overwritten in this case (default settings).

The ratio for the coupling defined via the NC parts program can only be input with language instruction COUPDEF (**...**).

The new ratio parameters take effect as soon as the COUPDEF instruction has been processed.

For further programming instructions for synchronous spindle couplings, please see Section 2.2.

**References**: /PGA/, "Programming Guide Advanced", Section 13.3

**Coupling properties**

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 ratio or the speed 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 in response to IPOSTOP (e.g. after setpoint-based 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:

  1. Setpoint coupling (DV)
     Application in position-controlled mode. The control dynamic response of both spindles should coincide as far as possible. The setpoint coupling should be used preferably.

  2. Actual-value coupling (AV)
     Application if position control of the LS is not 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.

  3. Speed coupling (VV)
     Internally, the speed 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 coupling characteristics are selected via machine data for fixed coupling configurations (see Section 2.3) and via language instruction COUPDEF for user-defined couplings (see Subsection 2.2.1).

In addition, coupling characteristics Type of coupling and Block change response can be altered for fixed coupling configurations by means of language instruction COUPDEF.

**Change protection for coupling characteristics**

Channel-specific MD 21340: COUPLE_IS_WRITE_PROT_1 is set to define whether or not the configured coupling parameters Speed ratio, Type of coupling and Block change response can be altered by the NC parts program:

0: Coupling parameters can be altered by the NC parts program via instruction COUPDEF

1: Coupling parameters cannot be altered by the NC parts program.
Attempts to make changes will be rejected with an alarm message.

**Overlaid movement**

In synchronous operation, the synchronous spindle copies the movement of the leading spindle in accordance with the programmed speed 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:

1. Programmable position offset of FS in AUTOMATIC and MDA:
   The position reference between LS and FS can be altered in active synchronous operation with language instructions COUPON and SPOS
   (see Subsection 2.1.2)
2. Manual position offset of FS:
   - In JOG operating mode (continuous JOG or incremental JOG):
     Overlay of FS by handwheel or plus or minus traversing key in active synchronous operation.
   - In AUTOMATIC and MDA operating modes:
     Overlay of FS by handwheel via 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 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 further information about specifying a positioning velocity using FA[Sn], please see

**References**: /FB/, S1, “Spindles”

---

## 2.1.2 Selecting synchronous mode

**Activation of coupling**

Language instruction COUPON activates the coupling 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 Subsection 2.1.4).

**Activation methods**

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 spindles. With this method, the angular offset must be programmed on selection.
   COUPON(FS, LS, $POS_{FS}$)

---

**Note**

If the LS and/or FS is in axis mode before switching on the synchronous coupling, the axis mode is left and spindle mode is activated with use of the spindle identifier with SW 3.2 and higher.

If the spindle is switched on with use of the axis identifier, no changeover takes place.

---

**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 Subsection 2.2.1).

**Determining current coupling status**

It is possible to determine the current coupling status for the specified axis/spindle in the NC parts program by means of axial system variable $AA_COUP_ACT[<axial expression >] (see Subsection 2.2.3 Axis system variables for synchronous spindles). 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 instructions 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 period 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 “Zero degrees position” of a position-controlled spindle is calculated from the zero mark signal or Bero signal of the measuring system and the offsets stored in axis-specific machine data (MD: REFP_SET_POS, REFP_MOVE_DIST, REFP_MOVE_DIST_CORR)

Range of $POS_{FS}$: 0 **...** 359.999 degrees.

**References:** /FB/, R1, “Reference Point Approach”

**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 parts program. The following two position offsets exist:

a) Current position offset of setpoint between FS and LS
$AA_COUP_OFFS [<axis identifier for FS>]

b) Current position offset of actual value between FS and LS
$VA_COUP_OFFS [<axis identifier for FS>]

(For more information about <axis identifier>, see Subsection 2.1.1)

**Activation after power ON**

Synchronous operation can also be activated for LS or FS which are not referenced/ synchronized (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.

## 2.1.3 Deselecting synchronous mode

**Deactivation of coupling**

Language instruction COUPOF cancels synchronous mode between the programmed spindles. The coupling concerned can be a fixed configuration 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, 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 deactivated, a block preprocessing stop (STOPRE) is generally initiated internally in the control.

**Deactivation while spindles are moving**

If synchronous mode is deselected while the spindles are in motion, 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 spindle can then be stopped from the parts program with M05, SPOS or SPOSA or from the PLC with the appropriate interface signal.

**Deselection methods**

Three different methods can be used to deselect synchronous mode:

1. Fastest possible deactivation of coupling.
   The block change is enabled immediately.
   COUPOF(FS, LS)
2. The 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. The coupling is not deselected until the following spindle and the leading spindle have crossed the programmed deactivation positions $POS_{FS}$ and $POS_{LS}$.
   The block change is then enabled.
   COUPOF(FS, LS, $POS_{FS}$ , $POS_{LS}$)

---

**Note**

If the LS and/or FS is in axis mode before switching off the synchronous coupling, the axis mode is left with use of the spindle identifier and the speed control mode is activated with SW 3.2 and higher.

If the spindle is switched off with use of the axis identifier, no changeover takes place. Before shutdown, the LS must be in the setpoint-side standstill.

---

**$POS_{FS}$, $POS_{LS}$**

Deactivation positions $POS_{FS}$ and $POS_{LS}$ match the actual positions of FS and LS respectively referred to the defined reference point value (see Subsection 2.1.2).

Range of $POS_{FS}$, $POS_{LS}$: 0 **...** 359.999 degrees.

**References:** /FB/, R1, “Reference Point Approach”

**COUPOFS and stopping the following spindle (SW version 6.4 and later)**

With SW version 6.4 and later, another deactivation method for a synchronous spindle coupling, i.e. by stopping the following spindle, has been added:

1. Fastest possible deactivation of coupling and
   stop without position specification.
   The block change is then enabled.
   COUPOFS(FS, LS)
2. A coupling is not deselected until the following spindle has crossed the deactivation position POS referred to the machine coordinate system. The block change is then enabled.
   COUPOFS(FS, LS, $POS_{FS}$)

**Supplementary condition:**
COUPOFS(FS, LS) and COUPOFS(FS, LS, $POS_{FS}$) have no meaning if a coupling was active.

### 2.1.4 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 a fixed configuration via machine data or according to user definition via parts program).
- The spindles to be coupled must be defined in the NC channel in which the coupling is activated.

  Channel-spec. MD 20070: AXCONF_MACHAX_USED
  axis-spec. MD 35000: 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
- LS and FS must be equipped with at least one position measuring system for position sensing.
- If the FS is in speed control mode before synchronous mode is activated (IS “Position controller active” DB31, ... DBX61.5 = 0), it must be switched to position control mode with the SPCON command.

---

**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).

---

After deactivation of synchronous operation, position control mode can be deselected again with language instruction SPCOF.

**References:** /FB/, S1, “Spindles”

- To ensure more accurate synchronization, the LS should be in position control mode (language instruction SPCON) before the coupling is activated, thus allowing a setpoint coupling to be established between the LS and FS.

  Actual-value coupling is always possible if there is a measuring system for the LS.
- 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.
- If FS and/or LS are in the axis mode and if they are actuated with a spindle identifier, spindle mode is activated. The VDI interface signals for the spindle concerned are modified, the active parameter block is changed over and feedforward control is activated.

  If the spindle is activated with use of the axis identifier, no changeover takes place.

**Cross-channel coupling (SW4)**

- The LS can be programmed either via a parts program, PLC (FC18) or, in SW 4 and higher, by means of synchronized actions.

---

**Note**

If the LS is swapped between channels with activated speed coupling, and the sequence of the channels is changed, the coupling must be deactivated.

**Example:**
Channel 1:
Channel 2:
Channel 3: FS in channel 3. COUPON active
Channel 4:
Channel 5:

Easy exchange possible for the LS between:
Channel 1 <—> Channel 2,
Channel 1 <—> Channel 3,
Channel 2 <—> Channel 3,
Channel 4 <—> Channel 5

Exchange possibilities for LS, where the coupling must be deactivated:
from Channel 1 <—> Channel 4,
from Channel 2 <—> Channel 4,
from Channel 3 <—> Channel 4,
from Channel 1 <—> Channel 5,
from Channel 2 <—> Channel 5,
from Channel 3 <—> Channel 5

---

**SW 5 and higher**

The LS can belong to 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 leading spindles can be overloaded.

### 2.1.5 Controlling synchronous spindle coupling via PLC (SW 6.3 and higher)

**Synchronous spindle extensions**

Using the coupling-specific, axial VDI interface signals, it is possible to control synchronization motions for the following spindle from the PLC. This offers the option of utilizing the PLC to disable, suppress, restore or modify a synchronization motion for the following spindle specified by offset programming.

These signals have no effect on the leading spindle. In SW 6.3 and earlier, 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 disabled from the PLC via axial interface signal "Disable synchronization" (DB31, ... DBX31.5).

When the main run advances to a block containing part program statement COUPON (FS, LS, offset), IS "Disable synchronization" (DB31, ... DBX31.5) is evaluated for the following spindle. With

- IS "Disable synchronization" (DB31, ... DBX31.5) = 0, the position offset is applied as before.
- IS "Disable synchronization" (DB31, ... DBX31.5) = 1, no additional following spindle motion takes place.

  The coupling responds analogously to a programmed COUPON (FS, LS) instruction without offset.

**Synchronized state reached**

Whenever a state of synchronism has been reached, the two VDI signals

IS "Coarse synchronism" (DB31, ... DBX98.1) and
IS "Fine synchronism" (DB31, ..., DBX98.0)

are set. Further block changes after COUPON are not prevented by suppression of synchronization.

**Example**

Block change behavior after COUPON

```
                                      ; Set IS "Disable synchronization"
                                      ; (DB31, ... DBX31.5) = 1 for S2

N51 SPOS=10 SPOS[2]=10                ; Positions correspond to an offset of 0 degrees
N52 COUPDEF(S2,S1,1,1,"FINE","DV")    ;
N53 COUPON(S2,S1,77)                  ; Actual offset of 0 degrees remains valid,
                                      ; no movement by following spindle, VDI signals
                                      , IS "Coarse synchronism" (DB31, ... DBX98.1)
                                      , and IS "Fine synchronism" (DB31, ... DBX98.0)
                                      ; are set and the block change is enabled.

N54 M0                                ;
N57 COUPOF(S2,S1)                     ;
N99 M30                               ;
```

**Reset and recovery**

Resetting IS “Disable synchronization” (DB31, ... DBX31.5) has no impact 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 can be recovered by repeating

part program statement COUPON (FS, LS, offset) with
IS “Disable synchronization” (DB31, ... DBX31.5) = 0

The parts program statement can be written, for example, in an ASUB.

**Read offset**

The following system variables can be used to read three different position offset values of the following spindle from the parts program.

| Description | NCK Variables |
|---|---|
| Programmed position offset of 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] |

**Special points to be noted**

Offset motion of the following spindle generated by

- SPOS, POS,
- Synchronized actions,
- FC18,
- JOG

cannot be controlled by IS “Disable synchronization” (DB31, ... DBX31.5). These functions are controlled by VDI signal IS “Feedrate stop/Spindle stop” (DB31, ... DBX4.3).

**“Feedrate stop/ spindle stop”**

Bit 4 in MD 30455: MISC_FUNCTION_MASK is configured to define the response of axial IS **“Feedrate stop/spindle stop”(DB31, ... DBX4.3)** for the following spindle.

Bit 4 is 0 Compatibility method
Canceling the feedrate enable brakes the coupled grouping.

Bit 4 is 1
Feedrate enable refers only to the interpolation component (SPOS),..) and does not affect the coupling.

---

**Note**

For further information about MD 30455: MISC_FUNCTION_MASK see:

**References**: /FB/, R2, “Rotary Axes”

---

### 2.1.6 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.

In this case, IS: “Fine synchronism” (DB31, ... DBX98.0) or “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 is 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 MD of the FS.

Reaching of the synchronism is influenced by the following factors:

- AV, DV: Position deviation between FS and LS
- VV: Speed difference between FS and LS

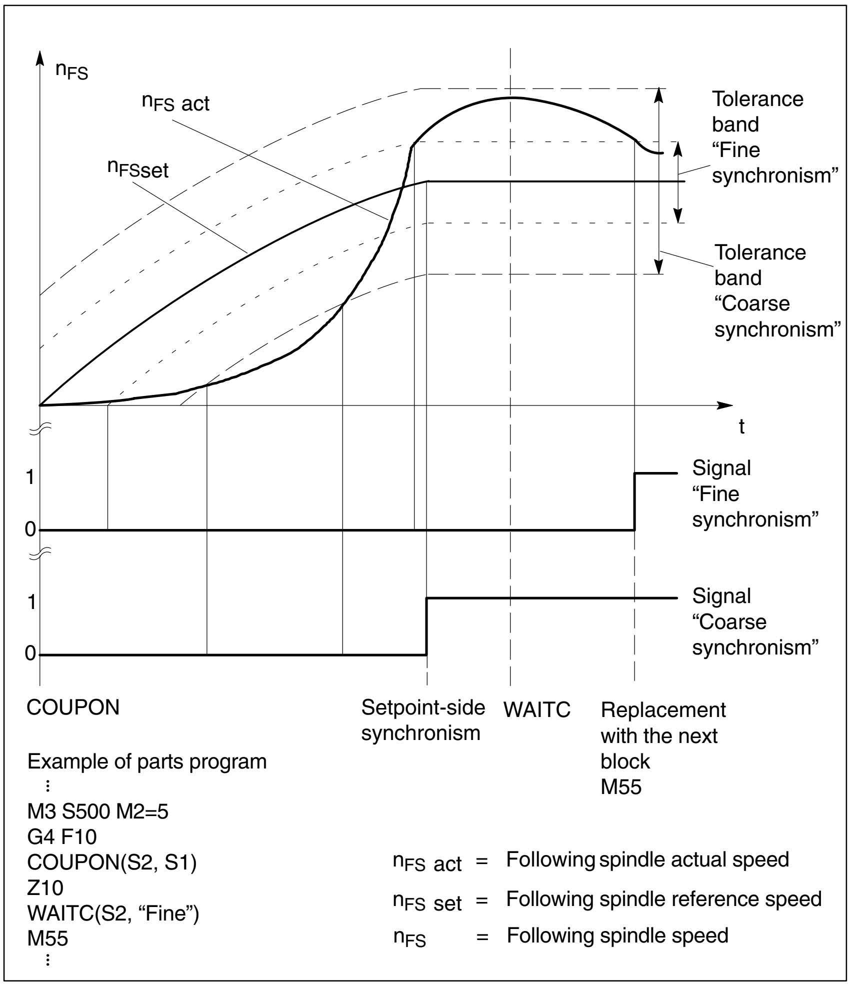



Fig. 2-1 Synchronism monitoring with COUPON and synchronism test marker WAITC

**Threshold values**

The relevant position or velocity tolerance range for the following spindle in relation to the leading spindle must be specified in degrees of rev/min.

- Threshold value for “Coarse synchronism”
  axis-spec. MD 37200: AV, DV: COUPLE_POS_TOL_COARSE
  MD 37220: VV: COUPLE_VELO_TOL_COARSE
- Threshold value for “Fine synchronism”
  axis-spec. MD 37210: AV, DV: COUPLE_POS_TOL_FINE
  MD 37230: 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.

## 2.2 Programming of synchronous spindle couplings

Table 2-1 Overview

<table>
<tr><th>Programmed coupling</th><th>Configured coupling(s)</th><th>Remarks</th></tr>
<tr><td>Definition of a coupling:<br>COUPDEF()</td><td>Modification of configured data:<br>COUPDEF()</td><td>Setting the coupling parameters</td></tr>
<tr><td colspan="2">Activating a coupling:<br>COUPON()</td><td rowspan="2">Activating<br>and deactivating</td></tr>
<tr><td colspan="2">Deactivating a coupling:<br>COUPOF()<br>by stopping the following spindle:<br>from SW 6.4 COUPOFS()</td></tr>
<tr><td>Deleting the coupling data:<br>COUPDEL()</td><td>Reactivating the configured data:<br>COUPRES()</td><td>Arrangement,<br>restore</td></tr>
</table>

**References:** /PGA/, Programming Guide Advanced "Synchronous Spindles"

### 2.2.1 Preparatory programming instructions

**User-defined coupling (SW 4 and lower)**

Up to two synchronous spindle couplings can be active simultaneously in each channel (SW 4 and lower). Provided no fixed coupling configuration has been programmed, both couplings can be freely defined by the NC parts program.

These couplings must also be parameterized by the NC parts program. Default values are used for parameters which are not programmed.

A new synchronous spindle coupling is defined if an FS/LS coupling relationship which has no fixed configuration is programmed in language instruction COUPDEF. This coupling can be invalidated again with language instruction COUPDEL if, for example, a further synchronous spindle coupling between other spindles is needed. These programming options, i.e. re-definition and deletion of couplings, allow more than two coupling relationships to be successively created in the NC channel (SW 4 and lower).

**SW 5 and higher**

Any number of couplings can be programmed. Furthermore, one coupling can also be configured via machine data as in earlier SW versions.

**Permanent coupling configuration**

The coupling characteristics and speed 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 instruction "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, $Ü_{numerator}$ , $Ü_{denominator}$ , block change response, coupling type)**

The spindle coupling is unambiguously defined with FS and LS FS and LS must be programmed in every COUP**...** statement otherwise alarm messages are generated.

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 identifiers for following and leading spindles

  E.g.: S1, SPI(1), S2, SPI(2)

  The applicable spindle number must be assigned to a machine axis in axis-specific
  MD: SPIND_ASSIGN_TO_MACHAX.

- $\mathbf{Ü_{numerator}}$ , $\mathbf{Ü_{denominator}}$ : Speed ratio parameters for numerator and denominator

  The speed ratio is input in the form of numeric values for numerator and denominator (see Subsection 2.1.1).

  The numerator must always be programmed. If no denominator is specified, then its value is always assumed to be "1,0".

- **Block change response**
  This parameter allows the condition for block change on selection of synchronous operation to be defined:

  **NO**C ⇒ Block change is immediately enabled
  **FI**NE ⇒ Block change in response to "Fine synchronism"
  **CO**ARSE ⇒ Block change in response to "Coarse synchronism"
  **IP**OSTOP ⇒ Block change in response to IPOSTOP (e.g. after setpoint-based synchronism)

  The block change response is entered 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 parts 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 marks **WAITC**, the replacement with new blocks is delayed until the parameterized synchronism is reached.

- **Type of coupling**

  **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 instruction “COUPDEL” is used to delete user-defined couplings.

**COUPDEL (FS, LS)**

**SW 4 and lower**

If a new synchronous spindle coupling relationship needs to be defined and all available, freely configurable couplings (1 or 2) are already configured, then one of the couplings will have to be deleted first.

**SW 5 and later**
There is no limit to the number of programmable couplings. The COUPDEL command can be used, but is not absolutely necessary.

**SW 6.3 and earlier**
An alarm message is generated if COUPDEL is programmed for an active coupling. Synchronous operation remains active. It must be deselected beforehand with COUPOF.

---

**Note**

A fixed coupling configuration cannot be deleted with COUPDEL in SW 6.3 and earlier!

**In SW 6.4 and later**, an active coupling deactivates this coupling, thereby deleting the coupling data. Alarm 16797 generated in SW 6.3 and earlier is now therefore irrelevant.
The following spindle rotates at the last valid speed. This corresponds to the behavior associated with COUPOF(FS, LS).

---

**Activate original coupling parameters**

Language instruction “COUPRES” can be used to re-activate the configured coupling parameters.

**COUPRES (FS, LS)**

The parameters programmed with COUPDEF (including speed ratio) are then overwritten.

Language instruction “COUPRES”
– activates the parameters stored in the machine and setting data (fixed coupling configuration) and
– activates the default settings (user-defined coupling).

**Programmable block change**

With SW 3.2 and higher, it is possible to mark a point in the NC program using “WAITC”. 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 Fig. 2-1).

**WAITC (FS)**

Advantage: The time between switching on the synchronous coupling and reaching synchronism can be technologically useful.

---

**Note**

Basically, it is always possible to write WAITC. If the spindle indicated is not active as FS, the instruction 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”)
```

## 2.2.2 Programming instructions for activating and deactivating the coupling

**Activate synchronous mode**

Language instruction 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.

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 degrees.

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)
```

### 2.2.3 Axial system variables for synchronous spindle

**Determining current coupling status**

The current coupling status for the following spindle can be read in the NC part program with the following axial system variable:

**$AA_COUP_ACT[<axial expression>]**

(For more information about <axial expression>, see Subsection 2.1.1)

**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

**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:

a) Setpoint-based position offset between FS and LS:

**$AA_COUP_OFFS[<axial expression>]**

b) 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 synchronism.

**Read programmed angular offset**

With SW version 6.3 and later, the last programmed position offset 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. In this case, the altered position offset can be read and corrected in the NC parts program if necessary.

---

## 2.2.4 Automatic selection and deselection of position control (SW 6.3 and later)

**Behavior in speed control mode**

With type of coupling DV, program instructions COUPON and COUPOF enable and disable the position control for the leading spindle as and when required. If several following spindles are coupled to the leading spindle, then the

- **first** DV coupling **activates** position control for the leading spindle and the
- **last** DV coupling **deactivates** position control for the leading spindle in speed control mode if SPCON is not programmed.

The leading spindle need not be configured in the same channel as the following spindle.

**Automatic selection with COUPON**

Depending on the coupling type, the effect of COUPON 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 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 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 specified position
Speed control mode is activated for the following spindle. Positioning mode is activated for COUPFS with stop position.
The position control is **not disabled** if the following spindle has been operating in position-controlled spindle mode with SPCON or COUPFS with position has been programmed.

[3] The position control is **disabled** by COUPOF if no other spindle is coupled to the leading spindle with coupling type DV.
Position control mode **remains active** if the leading spindle is operating in positioning or axis mode **or** has been operating in position-controlled spindle mode with SPCON.

## 2.3 Configuration of a synchronous spindle pair via machine data

**Coupling parameters**

**One** synchronous spindle coupling per NC channel can be configured permanently via channel-specific machine data.

It is then necessary to define the machine axes (spindles) which are to be coupled and what characteristics this coupling should have.

The following parameters can be configured as fixed settings for the synchronous spindle coupling:

- **Synchronous spindle pair** (channel-specific MD 21300: COUPLE_AXIS_1[n])

  This machine data defines the two machine axes which are to form the synchronous spindle pair (following spindle (n=0), leading spindle (n=1)).

  A 0 as the setting for the axis number means that no coupling is configured via the machine data. The machine data for the coupling characteristics are then irrelevant.

  The machine axis numbers for the LS and FS can not be changed by the NC parts program for a configured coupling configuration.

- **Speed ratio**

  This is entered via setting data using two ratio parameters (channel-spec. SD 42300: COUPLE_RATIO_1[n]) in the form of a numerator and a denominator. The quotient is generated internally in the control.

  $$k_{Ü} = \frac{\text{speed ratio parameter numerator}}{\text{speed ratio parameter denominator}} = \frac{\text{\$SC\_COUPLE\_RATIO[0]}}{\text{\$SC\_COUPLE\_RATIO[1]}}$$

  Provided it is not write-protected, the speed ratio can be changed by the NC parts program with language instruction COUPDEF.

- **Block change behavior**
  (channel-specific MD 21320: COUPLE_BLOCK_CHANGE_CTRL_1)

  One of the following options can be selected as the condition for a block change:

  0: Block changes immediately

  1: Block change in response to “Fine synchronism”

  2: Block change in response to “Coarse synchronism”

  3: Block change in response to IPOSTOP (i.e. after setpoint-based synchronism)

- **Type of coupling** between LS and FS:
  (channel-spec. MD 21310: COUPLING_MODE_1)

  0: Actual value coupling

  1: Setpoint coupling

  2: Speed coupling

- **Abortion of coupling with NC start**

  Channel-spec. MD 21330: COUPLE_RESET_MODE_1 (see Table 2-3)

- **Write-protection for coupling parameters**:

  (channel-spec. MD 21340: COUPLE_IS_WRITE_PROT_1)

  It can be defined in this machine data whether or not the configured coupling parameters Speed ratio, Type of coupling and Block change response may be influenced by the NC parts program.

  0: Coupling parameters can be changed by the NC parts program

  1: Coupling parameters cannot be changed by the NC parts program. Attempts to make changes are rejected with an alarm message.

### 2.3.1 Configuration of the behavior with NC start

The response to NC machining program start is defined by the channel-specific machine data.

Table 2-2 Synchronous coupling behavior with NC start

| | **Configured coupling** | **Programmed coupling (see Section 2.3)** |
|---|---|---|
| | MD: COUPLE_RESET_MODE | MD: START_MODE_MASK |
| Coupling is maintained | Bit 0 = 0 | Bit 10 = 0 |
| Deselect coupling | Bit 0 = 1 | Bit 10 = 1 |
| Activate configured data | Bit 5 = 1 | – |
| Activate coupling | Bit 9 = 1 | – |

### 2.3.2 Configuration of the behavior with Reset

With SW 3.2 and higher, the following behavior can be set with the channel-specific machine data with reset and end of NC machining program:

Table 2-3 Synchronous coupling behavior with end of NC machining program and after reset

| | **Configured coupling** | **Programmed coupling (see Section 2.3)** |
|---|---|---|
| Coupling is maintained | MD: COUPLE_RESET_MODE<br>Bit 1 = 0 | MD: RESET_MODE_MASK-<br>Bit 10 = 1 |
| Deselect coupling | MD: COUPLE_RESET_MODE<br>Bit 1 = 1<br>MD: RESET_MODE_MASK<br>Bit 0 = 1<br>(Generating a block on RESET) | MD: RESET_MODE_MASK-<br>Bit 10 = 0<br>Bit 0 = 1 |
| Activate configured data | MD: COUPLE_RESET_MODE<br>Bit 6 = 1<br>MD: RESET_MODE_MASK<br>Bit 0 = 1 | – |

# 2.4 Special features of synchronous operation

## 2.4.1 Special features of synchronous operation in general

**Control dynamics**

When a setpoint coupling is used, the position controller parameters of FS and LS (e.g. $K_V$ factor) must be matched. It may be necessary to activate different parameter sets for speed control mode and synchronous operation (M41...M45).

**Feedforward control**

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 MD 32620: FFW_MODE (=0). The NC parts 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 MD 32620: FFW_MODE.

**References:** /FB/, K3, "Compensation"

**Speed/acceleration limits**

The speed and acceleration limits of the spindles operating in synchronous mode are determined by the "weakest" spindle in the coupling. 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.

As an example, the maximum speed of the leading spindle is calculated internally in the control on the basis of 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 synchronous mode is activated, then the activation process is ignored and an alarm message generated.

Example of multiple couplings:

- A spindle is acting as the FS for several LS.

**As of SW 5.1**

**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.

**As of SW 5.2**

**Cross-channel setpoint linkage and optional number of following spindles in optional channels of an NCU:**

- Cross-channel synchronous spindle setpoint links (DV) can be implemented with no additional restrictions.
- Any number of following spindles in any channels on an NCU can be coupled to one leading spindle. The only possible restriction to the number of spindles could be imposed by the real CPU time requirement.

---

**Note**

The dynamic response of a coupling group 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 down to the load limit of the coupled leading spindles.

Further information: See above **Speed/acceleration limits**

---

**Knee-shaped acceleration characteristic**

The effect of a knee-shaped acceleration characteristic (identified by axis-specific MD 35220: ACCEL_REDUCTION_SPEED_POINT and MD 35230: ACCEL_REDUCTION_FACTOR) on the following spindle is taken into account for the leading spindle. If MD 35242: ACCEL_REDUCTION_TYPE is parameterized, it is also used in the configuration; otherwise, the reduction in acceleration is assumed to be hyperbolic.

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.

**As of SW 7.1**

**Knee-shaped acceleration characteristic for synchronous spindles**

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:** /FB/, B2, “Acceleration”, Knee-shaped acceleration characteristic

**Start synchronous mode using ASUB**

ASUBs (activation of asynchronous subprograms) processed by the PLC can be used to activate or terminate synchronous mode at any chosen time in the AUTOMATIC or MDA modes.

**References:** /FB/, K1, “Mode Group, Channels, Program Operation, Reset Response”

**Response to alarms**

If alarms (e.g. servo alarms), where the servo enable is canceled internally on the control, occur in synchronous mode and follow-up mode is active, the subsequent behavior is the same as if IS “Servo enable” (DB31, ... DBX2.1) had been canceled by the PLC (and “Follow-up mode” (DB31, ... DBX1.4) is set) → see Subsection 2.4.2 .

### 2.4.2 Influence on synchronous operation via PLC interface

**PLC interface signals**

In synchronous operation, the influence of the PLC on the coupling resulting from the setting of LS and FS interface signals must be noted.

The effect of the main PLC interface signals on the synchronous spindle coupling is described below.

**Spindle speed override (DB31, ... DBB19)**

The spindle speed override value input by the PLC in synchronous operation is applied only to the leading spindle.

**Axis/spindle disable (DB31, ... DBX1.3)**

The participating axes behave as shown in the following table (SW 4 and higher):

set: 1 not set: 0

| No. | LS/LA | FS/FA | Coupling | Procedure |
|---|---|---|---|---|
| 1 | 0 | 0 | Off | Setpoints of axes are output |
| 2 | 0 | 1 | Off | No setpoint output for FS/FA |
| 3 | 1 | 0 | Off | No setpoint output for LS/LA |
| 4 | 1 | 1 | Off | No setpoint output for LS/LA and FS/FA |
| 5 | 0 | 0 | ON | Setpoints of axes are output |
| 6 | 0 | 1 | ON | Disable not effective for FS/FA |
| 7 | 1 | 0 | ON | Disable also effective for FS/FA |
| 8 | 1 | 1 | ON | No setpoint output for LS/LA and FS/FA |

- This signal is no longer effective when the coupling for FS/FA is activated. → No. 6
- If the signal for the LS/LA is set, it also applies to the following spindle(s)/axis(es) →No. 7
- A workpiece clamped between two spindles (workpiece transfer from front to rearside machining) cannot be destroyed.

**Servo enable (DB31, ... DBX2.1)**

Cancellation of “Servo enable” for LS (either via PLC interface or internally in control in the event of faults):

If the servo enable signal of the LS is set to “0” during synchronous operation and a setpoint coupling is active, a switchover to actual-value coupling is executed in the control. If the LS is in motion at this instant, it is decelerated to a standstill and an alarm message generated. Synchronous operation remains active.

Cancellation of “Servo enable” for FS in synchronous operation (either via PLC interface or internally in control in the event of faults):

The coupling is internally canceled until the signals are reset.

If the “Servo enable” signal is not set for either of the spindles when synchronous operation is selected, synchronous operation is still activated when the coupling is switched on. The LS and FS however remain at standstill until the servo enable signal is set for both of them.

Setting the “Servo enable” signal for LS and FS:
When the signal edge of IS “Servo enable” switches to 1, the spindle either moves back to the old position (position on cancellation of servo enable) (signal status = 0: Stop active) or the current positions (position offset) are used again (signal status = 1: Follow-up active).

---

**Note**

If the “servo enable” signal is canceled for the FS after Spindle STOP without the coupling being deactivated beforehand, then any synchronism error resulting from external intervention (e.g. manual rotation) will not be compensated when the “servo enable” signal is applied again.

This may result in loss of the defined angular reference between the FS and LS for special applications.

---

**Followup mode (DB31, ... DBX1.4)**

Interface signal “Follow-up mode” is relevant only if the “servo enable” for the drive is canceled. When “servo enable” is set for the FS and LS, either the spindle will return to the position recorded on cancellation of the servo enable signal (signal state = 0: Stop active) or the current positions will be used again (signal status = 1: Follow-up active).

**Position measuring system 1/2 (DB31, ... DBX1.5 and 1.6)**

Switchover between the position measuring systems for the FS and LS is not locked out in synchronous operation. A switchover would not affect the coupling. It is however recommended that the measuring systems only be switched when synchronous mode is not active.

If “Park” status is selected for the FS or LS in synchronous operation, then the system responds as if “servo enable” had been canceled.

**Delete distance to go/spindle reset (DB31, ... DBX2.2)**

When Spindle RESET is set for the LS in synchronous operation, the LS is braked down to standstill at the selected acceleration rate. The FS and LS continue to operate in synchronous mode. The overlaid motion (except with COUP...) is terminated as quickly as possible.

**Spindle stop (Feed stop) (DB31, ... DBX4.3)**

When “Spindle STOP” is set for the FS or LS, both coupled spindles are braked down to standstill via a ramp, but continue to operate in synchronous mode.

As soon as IS “Spindle STOP” is no longer active for any of the spindles in the coupling, it is accelerated back up to the previous speed setpoint.

Application

“Spindle STOP” can halt the synchronous spindle pair without offset since the servo loop remains operative.

Example

When the protective door is opened with an active synchronous spindle coupling, the FS and LS must be stopped without the coupling relationship being altered. This can be achieved by applying IS “Spindle stop” to halt the FS and LS (IS “Axis/spindle stationary” (DB31, ... DBX61.4) = 1). “Servo enable” can then be canceled for both spindles.

**Delete S value (DB31, ... DBX16.7)**

The S value programmed for the LS is deleted and the LS decelerated down to zero speed via a ramp. The FS and LS continue to operate in synchronous mode.

IS “Delete S value” has no affect on the FS in synchronous operation.

**Resynchronize spindle 1/2 (DB31, ... DBX16.4 and 16.5)**

It is possible to synchronize the spindle (LS) with its positioning measuring system when it is operating in synchronous mode. It is however recommended that the leading spindle only be re-synchronized when synchronous mode is not active.

**Traverse keys for JOG (DB31, ... DBX4.6 and 4.7)**

The “plus and minus traversing keys” for JOG are **not disabled** internally for the FS in synchronous operation, i.e. the FS executes an overlaid motion if one of these keys is pressed.

---

**Note**

If overlaid 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 Subsection 2.3.1.

---

**Note**

NC Start after NC Stop does not deselect synchronous operation.

---

### 2.4.3 Differential speed between leading and following spindles (SW version 7.1 and later)

**When does a differential speed occur?**

A differential speed develops, e.g. with turning machine applications, when two spindles are opposite. 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 a

- speed with S... and a
- 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.

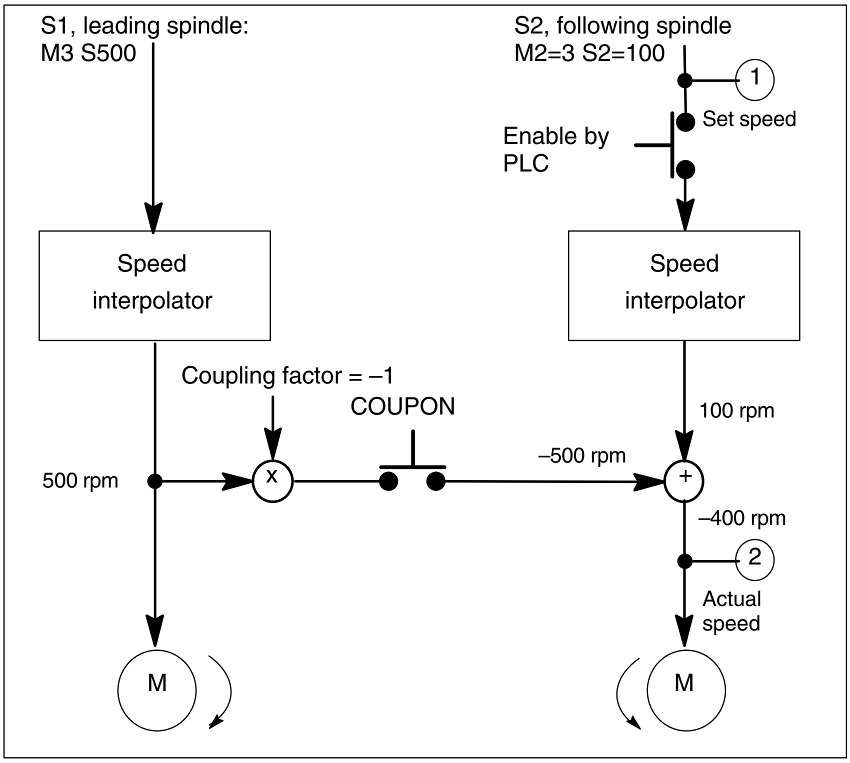



Fig. 2-2 Schematic representation of process resulting in differential speed

**Example**

```
N01 M3 S500             ; S1 rotates 500 rev/min in positive direction, spindle 1 is
                        ; master spindle
N02 M2=3 S2=300         ; S2 rotates 300 rev/min in positive direction
...
N05 G4 F1
N10 COUPDEF(S2,S1,–1)   ; Coupling factor –1:1
N11 COUPON(S2,S1)       ; Activate coupling, speed of following spindle S2 is calculated
...                     ; from speed for leading spindle S1 and coupling factor.
N26 M2=3 S2=100         ; Program differential speed, S2 is following spindle
```

**Application**

Manufacturing operations with positioned leading spindle and rotating tools require exact synchronism with the counterspindle 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.

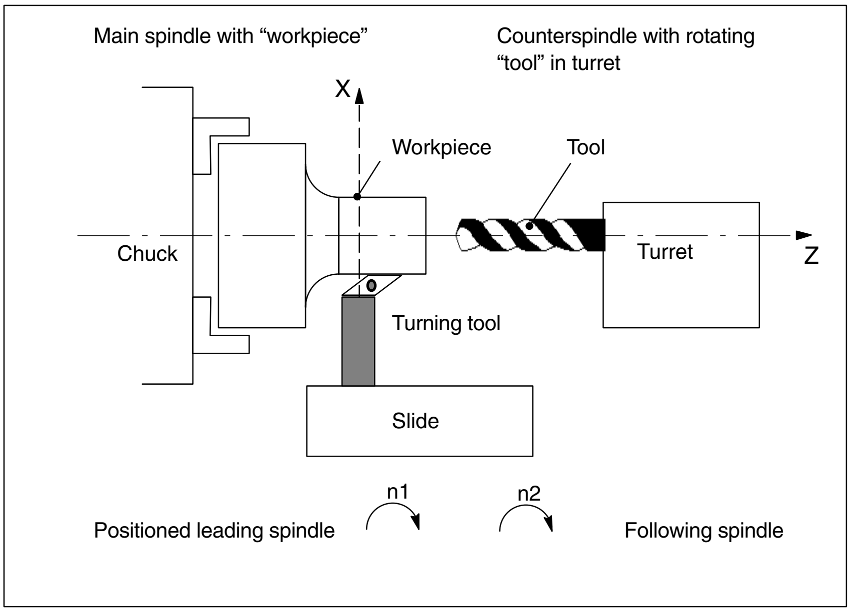



Fig. 2-3 Application on a single-slide turning machine with turret about Z axis

**Behavior during and after coupling**

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.

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.

---

**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.

---

**Differential speed**

A differential speed occurs as a result of **re-**programming the following spindle (S2=... in the example) or M2=3, M2=4 in speed control mode **during** an active synchronous spindle coupling.

**Supplementary condition**
Speed S... must also be programmed again with direction of rotation M3 or M4. Otherwise alarm 16111 “Channel% Block% Spindle% No speed programmed” is displayed.

**Prerequisites**

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. The system may otherwise react unpredictably in, for example, rigid tapping operations 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.

**Read offsets of following spindle**

The current offset always changes when a differential speed is programmed. The current position offset is read with $AA_COUP_OFFS[Sn] at the setpoint end and with $VA_COUP_OFFS[Sn] at the actual value end.
The last programmed offset is supplied by variable $P_COUP_OFFS[Sn].

When a differential speed is programmed (equals 100 rev/min in the example), the programmed differential component is displayed as the speed setpoint. The actual speed value refers to the motor speed. In the example, the actual speed is 500 rev/min * (–1) + 100 rpm = –400 rev/min.

**NCK to PLC**

**Following spindle in speed control mode**
IS "Spindle in setpoint range (DB31, ... DBX83.5) is set by the NCK for the following spindle if the programmed speed (see example above N26 with M2=3 S2=100) is reached at a differential speed of 100 rev/min. 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**

At the output end, axial VDI interface signals NCK → PLC IS "Coarse/fine synchronism" (DB31, ... DBX98.1/98.0) are reset and IS "Overlaid movement" (DB31, ... DBX98.4) is set if setpoints are generated, in addition to the coupling setpoints, as a result of programming the differential speed.

---

**PLC to NCK**

**Manipulation of the following spindle via the PLC interface**
The effect of the axial VDI interface signals on the following spindle with a differential speed in speed control mode is described below:

**Actual direction of rotation CW (DB31, ... DBX83.7)**

IS "Actual direction of rotation CW" (DB31, ... DBX83.7) refers to the resulting motor direction.

**Delete distance to go/spindle reset (DB31, ... DBX2.2)**

The programmed differential speed and direction can be terminated by IS "Delete distance to go "Reset spindle" (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)**

IS "Resynchronize spindle 1/2" (DB31, ... DBX16.4/16.5) are **not** interlocked. 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 programmed speed component for the following spindle.

The motion component generated by the synchronous spindle coupling remains unaffected.

**Spindle speed override (DB31, ... DBB19)**

The "Spindle speed override" VDI interface (DB31, ... DBB19) only acts on the 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 **double** to the following spindle.

Once
– indirectly by a change in the leading spindle speed 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 on coupling deselection is at continuous speed.

When COUPOF is programmed, the spindle behaves as if it had been programmed with the transferred speed and rotational direction.
This equals M4 S400 in the example.

When COUOPOFS 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. When G96 S2=... the "constant cutting rate" function can be activated for the following spindle.
  The calculated speed equals the set speed for the speed interpolator of spindle 2 and is thus added to the total speed for S2.
- Rigid tapping with G331, G332.

### 2.4.4 Restore synchronism of following spindle (SW 7.1 and later)

**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 has occurred (internal follow-up mode, IS "Follow-up mode active" (DB31, ... DBX61.3) = 1). When the alarm is cleared, the NC must not trigger any synchronization motion.
- The spindles have not been synchronized due to a synchronization disable caused by following spindle IS "Disable synchronization" (DB31, ... DBX29.5).

**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 resynchronization process with VDI signal "Synchronize following spindle". Only when this process is complete can setpoint synchronism be fully restored.
3. Wait until the actual values of the coupled spindles have synchronized.

**Enable resynchronization**

Setting the enabling signals closes the coupling at the current actual positions. IS "Coarse synchronism" (DB31, ... DBX98.1) and IS "Fine synchronism" (DB31, ..., DBX98.0) continue to refer to the last coupling parameters and are not set automatically. The following **preconditions** must be fulfilled for resynchronization to work:

- The axis enabling signal must be set for the following spindle.
- No synchronization disable signal
  IS "Disable synchronization" (DB31, ... DBX31.5) must be set for the following spindle.

**Resynchronize following spindle**

Resynchronization is started explicitly for the relevant following spindle and commences as soon as the low-high edge of axial

**IS "Synchronize following spindle"(DB31, ... DBX31.4** ) is detected.

The NC acknowledges detection of the signal edge by output of axial IS "Synchronization in progress" (DB31, ... DBX99.4) for the following spindle and the signal remains set for as long as IS "Synchronize following spindle" (DB31, ... DBX31.4) is set.

If a synchronization already exists, axial IS "Synchronization in progress (DB31, ... DBX99.4) remains active at least until synchronism between following and leading spindle has been established at the setpoint end.

**Determine synchronous spindle position**

The spindle position to be synchronized is determined by the programmed offset position between the following and leading spindles, e.g. COUPON(...,77).

If the “Correct synchronism” function (see below) detects a positional difference, this is also taken into account.

**Correct and restore synchronism**

The following and leading spindles are always synchronized as quickly as possible. IS “Overlaid movement” (DB31, ... DBX98.4) and IS “Synchronization in progress” (DB31, ... DBX99.4) are output for the following spindle while the synchronization setpoints are being generated.

Synchronism is not established **at the setpoint end** until the two signals IS “Overlaid movement” (DB31, ... DBX98.4) and IS “Synchronization in progress” (DB31, ... DBX99.4) have been canceled.
The length of time which elapses before the two signals IS “Coarse synchronism” (DB31, ... DBX98.1) and IS “Fine synchronism” (DB31, ... DBX98.0) occur for synchronization **at the actual value end** depends mainly on the dynamic response of the drives involved in the coupling.

**Example**

```
N51 SPOS=0 SPOS[2]=90
N52 COUPDEF(S2,S1,1,1,“FINE”,“DV”)
N53 COUPON(S2,S1,77)
N54 M0                         ; Cancel servo enable,
                               ; Set correction, rotate following
                               ; spindle backwards by 11 degrees
```

The system variables return the following values for the following spindle:

```
$P_COUP_OFFS[S2]               ; Programmed position offset = 77 degrees
$AA_COUP_OFFS[S2]              ; Position offset in setpoints = 66 degrees
$VA_COUP_OFFS[S2]              ; iPosition offset in actual values approx. 66
                               ; degrees
```

The synchronism signals refer to the programmed position offset of 77 degrees and would no longer be set if there were a synchronism tolerance of 0.5 or 2 degrees, as the deviation is about 11 degrees.

**Overlaid motion**
An overlaid movement on the following spindle is always indicated by IS “Overlaid movement” (DB31, ... DBX98.4).
This additional movement can be generated by SPOS, M3 S... , JOG, positioning via FC18 or synchronized actions.

---

**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 “Synchronize following spindle” (DB31, ... DBX31.4), but is restored by the REPOS operation.

---

**Stop and block change**

If “Stop” was activated during cancelation of the axis enables for the leading or following spindle, the **last** position setpoints are approached again when the axes are enabled by the servo drive.

Program instructions COUPON and WAITC can influence the block change behavior. In this case, the block change criterion is defined by COUPDEF or via MD 21320: COUPLE_BLOCK_CHANGE_CTRL_1.

### 2.4.5 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. This start-up procedure is described in:

**References:** /IAD/, SINUMERIK 840D Installation and Start-Up Guide
and
**References**: /FB/, S1, “Spindles”

**Required**

The following parameters must then be set for the synchronous spindle pair:

- The machine axis numbers for the leading and following spindles (for a permanently configured coupling with channel-spec. MD: COUPLE_AXIS_1[n])
- The required coupling type (setpoint, actual-value or velocity coupling) (for a permanently configured coupling with channel-spec. MD: COUPLING_MODE_1[n])
- The gear stage(s) of FS and LS for synchronous operation
- Plus the following coupling properties (see Section 4.1) for a permanently configured synchronous spindle coupling:
  - Block change behavior in synchronous spindle operation
    Channel-spec. MD: COUPLE_BLOCK_CHANGE_CTRL_1
  - Coupling abort behavior
    Channel-spec. MD: COUPLE_RESET_MODE_1
  - Modification protection for coupling parameters
    Channel-spec. MD: COUPLE_IS_WRITE_PROT_1
  - Speed ratio parameters for synchronous spindle coupling
    Channel-spec. SD: COUPLE_RATIO_1[n].

**Response to setpoint changes**

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

The following control parameters must each be set optimally for the FS and LS:

- $K_V$ factor (MD 32200 POSCTRL_GAIN)
- Feedforward control parameters
  MD 32620 FFW_MODE
  MD 32610 VELO_FFW_WEIGHT
  MD 32650 AX_INERTIA
  MD 32800 EQUIV_CURRCTRL_TIME
  MD 32810 EQUIV_SPEEDCTRL_TIME

**References:** /FB/, K3, “Compensation”

The following control parameters must be set identically for the FS and LS:

- Fine interpolator type (MD 33000: FIPO_TYPE)
- Axial jerk limitation
  MD 32400 AX_JERK_ENABLE
  MD 32410 AX_JERK_TIME
  MD 32420 JOG_AND_POS_JERK_ENABLE
  MD 32430 JOG_AND_POS_MAX_JERK

**References:** /FB/, G2, “Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control”

**Dynamic response adaptation**

The FS and the coupled LS must have the same dynamic response to setpoint changes. The same dynamic response means that their following errors must be equal at any given speed.

The dynamic response adaptation function in the setpoint branch makes it possible to obtain an excellent match in the response to setpoint changes between axes, which have different dynamic characteristics (control loops). The difference in the equivalent time constants between the dynamically “weakest” spindle and the other spindle in the coupling must be entered as the dynamic response adaptation time constant.

Example

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

Leading spindle: MD 32810: EQUIV_SPEEDCTRL_TIME [n] = 5ms

Following spindle: MD 32810: EQUIV_SPEEDCTRL_TIME [n] = 3ms

→ Time constant of dynamic response adaptation for following spindle:
MD 32910: DYN_MATCH_TIME [n] = 5 ms – 3 ms = 2 ms

The dynamic response adaptation must be activated axially via MD 32900 DYN_MATCH_ENABLE.

The dynamic adaptation setting can be checked by comparing the following errors of the FS and LS (in Diagnosis operating area; Service Axes display). Their following errors must be identical when they are operating at the same speed!

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

**Control parameter sets**

A separate parameter set with servo loop setting is assigned to each gear stage on coupled spindles.

These parameter sets can be used, for example, to adapt the dynamic response of the leading spindle to the following spindle in synchronous operation. When the coupling is deactivated (speed or positioning mode), it is therefore possible to select other position controller parameters for the FS and LS. To utilize this option, a separate gear stage must be reserved for synchronous operation and selected before synchronous mode is activated.

**References:** /FB/, G2, “Velocities, Setpoint/ActualValue Systems, ClosedLoop Control”

**Actual value coupling**

In an actual-value coupling, 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 corresponds internally to a setpoint coupling, but with lower dynamic requirements of the FS and LS. A servo loop is not needed for the FS and/or LS and no measuring systems are needed.

**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-spec. MD 37200: AV, DV: COUPLE_POS_TOL_COARSE
  MD 37220: VV: COUPLE_VELO_TOL_COARSE
- Threshold value for “Fine synchronism”
  axis-spec. MD 37210: AV, DV: COUPLE_POS_TOL_FINE
  MD 37230: VV: COUPLE_VELO_TOL_FINE

The values must be calculated according to the accuracy requirements of the machine manufacturer (check via the PLC interface or in the FS 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 selected, the “zero degree positions” of the FS and LS must be mutually adapted. This can be done with the following machine data:

MD 34100 REFP_SET_POS
MD 34080 REFP_MOVE_DIST
MD 34090 REFP_ MOVE_DIST_CORR

**References:** /FB/, R1, “Reference Point Approach”

**Service display for FS**

The following values are displayed for the following spindle for start-up in synchronous operation in the “Service Values Axes” display in the “Diagnosis” operating area:

- Actual deviation between setpoints of FS and LS
  Display value: 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 parts program)
- Actual deviation between actual values of FS and LS
  Display value: Position offset in relation to leading spindle (actual value)

■

# Notes

# 3 Supplementary Conditions

**Availability of "Synchronous spindle" function**

The function is an option and is available for

- SINUMERIK 840D, SW 2 and higher
- SINUMERIK 810D, CCU2, SW 2 and higher

**Availability of " WAITC" function**

This function is available together with synchronous spindle for

- SINUMERIK 840D, SW 3 and higher

■

# 4 Data Descriptions (MD, SD)

## 4.1 Description of machine data

### 4.1.1 Channelspecific machine data

| | | | |
|---|---|---|---|
| **21300**<br>MD number | **COUPLE_AXIS_1[n]**<br>Definition of synchronous spindle pair [n] | | |
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 8 | |
| Changes effective after POWER ON | Protection level: 2/7 | Unit: – | |
| Data type: BYTE | Applies from SW: 2.1 | | |
| Meaning: | **One** synchronous spindle pair per NC channel can be defined in a **fixed configuration** with this machine data.<br><br>The machine axis numbers (channel-specific MD: AXCONF_MACHAX_USED) applicable in the NC channel must be entered for the following spindle [n=0] and the leading spindle [n=1]. MD: AXCONF_MACHAX_USED).<br><br>If a value of "0" is entered, then the coupling is not configured, thus leaving 2 couplings to be configured freely via the NC parts program. | | |
| MD irrelevant for ... ... | User-defined coupling | | |
| Related to .... | Channel-spec. MD: COUPLING_MODE_1 (type of coupling in synchronous spindle mode)<br>Channel-spec. MD: COUPLE_IS_WRITE_PROT_1 (write-protection for coupling parameters)<br>Channel-spec. MD: COUPLE_RESET_MODE_1 (coupling abortion response)<br>Channel-spec. MD: COUPLE_BLOCK_CHANGE_CTRL_1 (block change response in synchronous spindle mode)<br>SD: $SC_COUPLE_RATIO_1 (speed ratio parameters for synchronous spindle mode) | | |

| | | | | | |
|---|---|---|---|---|---|
| **21310**<br>MD number | **COUPLING_MODE_1**<br>Type of coupling in synchronous spindle mode | | | | |
| Default setting: 1 | | Minimum input limit: 0 | | Maximum input limit: 2 | |
| Changes effective after POWER ON | | | Protection level: 2/7 | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 2.1 | | |
| Meaning: | This machine data determines the type of coupling for the fixed coupling configuration defined with machine data COUPLE_AXIS_1[n].<br><br>1: Setpoint coupling activated.<br>With a setpoint coupling, the reference value for the following spindle is calculated from the position setpoint of the leading spindle, allowing the setpoints for the FS and LS to be input simultaneously. This has a particularly positive effect on the spindle synchronism during acceleration and deceleration processes.<br><br>A better response to setpoint changes is thus obtained with the setpoint coupling than with the actual-value coupling.<br><br>When a setpoint coupling is selected, the following conditions must be fulfilled before synchronous operation is activated:<br>• The LS must be assigned to the same NC channel as the FS.<br>• The FS and LS must be in position control mode (SPCON)<br>• The FS and LS must have the same dynamic control response (see Subsection 2.4.3)<br><br>0: Actual-value coupling activated.<br>With a setpoint coupling, the reference value for the following spindle is calculated from the position setpoint of the leading spindle, With this type of coupling, the following drive must be significantly more dynamic than the leading drive, but never vice versa.<br><br>The actual-value coupling can be used, for example, in the following applications:<br>• The LS must be assigned to a different NC channel than the FS<br>• For leading spindles which are not suitable for position control<br>• In cases where the dynamic control response of the leading spindle is considerably slower than that of the following spindle.<br><br>As soon as the actual-value coupling is active, the IS “Actual-value coupling” for the FS is set to “1”.<br><br>2: Velocity coupling activated.<br>Internally, the velocity coupling is a setpoint coupling. The requirements placed on FS and LS are lower. A defined position relation between FS and LS cannot be established.<br><br>In the following cases, the velocity coupling is applied:<br>• LS and/or FS are not in position control.<br>• There are no measuring systems.<br><br>The coupling type can be altered in the NC part program when the coupling is deactivated by means of language instruction COUPDEF provided that this option is not inhibited in channel-specific MD: COUPLE_IS_WRITE_PROT_1. The parameterized value of channel-specific MD: COUPLING_MODE_1 does not, however, get altered. | | | | | |
| MD irrelevant for ... ... | User-defined coupling | | | | |
| Related to .... | Channel-spec. MD: COUPLE_AXIS_1 (definition of synchronous spindle pair)<br>Channel-specific MD: COUPLE_IS_WRITE_PROT_1 (write-protection for configured parameters)<br>IS “Actual-value coupling” (DB31-48, DBX98.2) | | | | |

| 21320<br>MD number | COUPLE_BLOCK_CHANGE_CTRL_1<br>Block change response in synchronous spindle mode | | |
|---|---|---|---|
| Default setting: 3 | Minimum input limit: 0 | Maximum input limit: 3 | |
| Changes effective after POWER ON | Protection level: 2/7 | Unit: – | |
| Data type: BYTE | Applies from SW: 2.1 | | |
| Meaning: | This machine data determines the condition on which a block change must be executed when synchronous mode is activated for the fixed coupling configuration defined in channel-specific machine data COUPLE_AXIS_1[n].<br><br>The following options are available:<br><br>0: Block change is enabled immediately<br>1: Block change in response to “Fine synchronism”<br>2: Block change in response to “Coarse synchronism”<br>3: Block change in response to IPOSTOP (i.e. after setpoint-based synchronism)<br><br>The block change response can be altered in the NC part program with language instructions COUPDEF provided this option has not been inhibited with channel-specific MD: COUPLE_IS_WRITE_PROT_1. The parameterized value of channel-specific MD: COUPLE_BLOCK_CHANGE_CTRL_1 does not, however, get altered.<br><br>The selected block change response remains valid even when the speed ratio is changed or a defined angular offset is programmed while the coupling is active. | | |
| MD irrelevant for ... ... | User-defined coupling | | |
| Related to .... | Channel-spec. MD: COUPLE_AXIS_1 (definition of synchronous spindle pair)<br>Channel-specific MD: COUPLE_IS_WRITE_PROT_1<br>(write-protection for coupling parameters)<br>Channel-spec. MD: COUPLE_POS_TOL_COARSE or COUPLE_VELO_TOL_COARSE<br>(threshold value for coarse synchronism)<br>Channel-specific MD: COUPLE_POS_TOL_FINE or COUPLE_VELO_TOL_FINE<br>(threshold value for fine synchronism) | | |

| 21330<br>MD number | COUPLE_RESET_MODE_1<br>Coupling abort response | | |
|---|---|---|---|
| Default setting: 1 | Minimum input limit: 0 | Maximum input limit: 0x3FF | |
| Changes effective after POWER ON | Protection level: 2/7 | Unit: – | |
| Data type: BYTE | Applies from SW: 2.1 | | |
| Meaning: | The behavior of synchronism for the synchronous spindle pair configured with the machine data COUPLE_AXIS_1[n] is defined with this machine data.<br><br>Bit 1=0: Synchronism remains active even the program is started again and can be canceled only with COUPOF as long as the control remains switched on.<br><br>Bit 0=1: Synchronism is canceled with program start (from the reset condition).<br><br>Bit 1=0: Synchronism remains active even with program end and RESET and can be canceled only with COUPOF as long as the control remains switched on.<br><br>Bit 1=1: Synchronism is canceled with program end or RESET.<br><br>Bit 5=1: The configured data are activated with program start.<br><br>Bit 6=1: The configured data are activated with program end or RESET.<br><br>Bit 9=1: Synchronism is switched on with program start.<br><br>Note: Synchronism is not deselected with NC start after NC stop! | | |
| MD irrelevant for ... ... | User-defined coupling | | |
| Related to .... | Channel-spec. MD: COUPLE_AXIS_1 (definition of synchronous spindle pair)<br>IS “Synchronous operation” (DB31-48, DBX84.4) | | |

| | | | | | |
|---|---|---|---|---|---|
| **21340**<br>MD number | **COUPLE_IS_WRITE_PROT_1**<br>Coupling parameters are write-protected | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: 2/7 | | Unit: – |
| Data type: BOOLEAN | | | | Applies from SW: 2.1 | |
| Meaning: | This machine data is used to specify whether or not the coupling parameters (speed ratio, block change response, coupling type) for the synchronous spindle pair configured with channel-specific machine data COUPLE_AXIS_1[n] may be altered by the NC part program.<br>1: Coupling parameters may **not** be altered by the NC program (write-protection active). An alarm message is generated if an attempt is made to change the parameters.<br>0: NC parts program may alter coupling parameters using language instructions COUPDEF. | | | | |
| MD irrelevant for ... ... | User-defined coupling | | | | |
| Related to .... | Channel-spec. MD: COUPLE_AXIS_1 (definition of synchronous spindle pair)<br>Channel-specific MD: COUPLING_MODE_1 (type of coupling in synchronous spindle mode)<br>Channel-spec. MD: COUPLE_RESET_MODE_1 (coupling abortion response)<br>Channel-spec. MD: COUPLE_BLOCK_CHANGE_CTRL_1 (block change response in synchronous spindle mode)<br>SD: $SC_COUPLE_RATIO_1 (speed ratio parameters for synchronous spindle mode) | | | | |

### 4.1.2 Axis-specific machine data

| | | | | | |
|---|---|---|---|---|---|
| **37200**<br>MD number | **COUPLE_POS_TOL_COARSE**<br>Threshold value for coarse synchronism | | | | |
| Default setting: 1.0 | | Minimum input limit: 0.0 | | Maximum input limit: PLUS | |
| Changes effective after NEW_CONF | | | Protection level: 2/7 | | Unit:<br>Linear axis: mm<br>Rotary axis: Degrees |
| Data type: DOUBLE | | | | Applies from SW: 2.1 | |
| Meaning: | In synchronous operation, the positional deviation between the leading and following spindles is monitored (only DV and AV mode).<br>IS “Coarse synchronism” is set if the current positional deviation is within the tolerance band specified by the threshold value.<br>Furthermore, this threshold value represents the criterion for a block change on activation of synchronous operation or on alteration of the transmission parameters when the coupling is active in cases where “Coarse synchronism” is selected as the block change response condition (see channel-specific MD: COUPLE_BLOCK_CHANGE_CTRL_1 or language instruction COUPDEF).<br>If the value “0” is input, IS “Coarse synchronism” is always set to “1” in DV and AV mode. | | | | |
| Related to .... | Channel-spec. MD: COUPLE_BLOCK_CHANGE_CTRL_1 (block change response in synchronous spindle operation)<br>IS “Coarse synchronism” (DB31-48, DBX98.1) | | | | |

| | | | | |
|---|---|---|---|---|
| **37210**<br>MD number | **COUPLE_POS_TOL_FINE**<br>Threshold value for fine synchronism | | | |
| Default setting: 0.5 | | Minimum input limit: 0.0 | | Maximum input limit: PLUS |
| Changes effective after NEW_CONF | | Protection level: 2/7 | | Unit:<br>Linear axis: mm<br>Rotary axis: Degrees |
| Data type: DOUBLE | | | Applies from SW: 2.1 | |
| Meaning: | In synchronous operation, the positional deviation between the leading and following spindles is monitored (only DV and AV mode).<br>IS “Fine synchronism” is set if the current positional deviation is within the tolerance band specified by the threshold value.<br>Furthermore, this threshold value represents the criterion for a block change on activation of synchronous operation or on alteration of the transmission parameters when the coupling is active in cases where “Fine synchronism” is selected as the block change response condition (see channel-specific MD: COUPLE_BLOCK_CHANGE_CTRL_1 or language instruction COUPDEF).<br>If the value “0” is input, IS “Fine synchronism” is always set to “1” in DV and AV mode. | | | |
| Related to .... | Channel-spec. MD: COUPLE_BLOCK_CHANGE_CTRL_1 (block change response in synchronous spindle operation)<br>IS “Fine synchronism” (DB31-48, DBX98.0) | | | |

| | | | | |
|---|---|---|---|---|
| **37220**<br>MD number | **COUPLE_VELO_TOL_COARSE**<br>“Coarse” speed tolerance between leading and following spindles | | | |
| Default setting: 1.0 | | Minimum input limit: 0.0 | | Maximum input limit: PLUS |
| Changes effective after NEW_CONF | | Protection level: 2/7 | | Unit:<br>Linear axis: mm/min<br>Rotary axis: rev/min |
| Data type: DOUBLE | | | Applies from SW: 3.1 | |
| Meaning: | In synchronous operation, the speed difference between the leading and following spindles is monitored (VV mode only).<br>IS “Coarse synchronism” is set if the current speed difference is within the tolerance band specified by the threshold value.<br>Furthermore, this threshold value represents the criterion for a block change on activation of synchronous operation or on alteration of the transmission parameters when the coupling is active in cases where “Coarse synchronism” is selected as the block change response condition (see channel specific MD: COUPLE_BLOCK_CHANGE_CTRL_1 or language instruction COUPDEF).<br>If the value “0” is input, IS “Coarse synchronism” is always set to “1” in VV mode. | | | |
| Related to .... | Channel-spec. MD: COUPLE_BLOCK_CHANGE_CTRL_1 (block change response in synchronous spindle operation)<br>IS “Coarse synchronism” (DB31-48, DBX98.1) | | | |

| | | |
|---|---|---|
| **37230**<br>MD number | **COUPLE_POS_TOL_FINE**<br>“Fine” speed tolerance between leading and following spindles | |
| Default setting: 0.5 | Minimum input limit: 0.0 | Maximum input limit: PLUS |
| Changes effective after NEW_CONF | Protection level: 2/7 | Unit:<br>Linear axis: mm/min<br>Rotary axis: rev/min |
| Data type: DOUBLE | Applies from SW: 3.1 | |
| Meaning: | In synchronous operation, the speed difference between the leading and following spindles is monitored (VV mode only).<br>IS “Fine synchronism” is set if the current speed difference is within the tolerance band specified by the threshold value.<br>Furthermore, this threshold value represents the criterion for a block change on activation of synchronous operation or on alteration of the transmission parameters when the coupling is active in cases where “Fine synchronism” is selected as the block change response condition (see channel-specific MD: COUPLE_BLOCK_CHANGE_CTRL_1 or language instruction COUPDEF).<br>If the value “0” is input, IS “Fine synchronism” is always set to “1” in VV mode. | |
| Related to .... | Channel-spec. MD: COUPLE_BLOCK_CHANGE_CTRL_1 (block change response in synchronous spindle operation)<br>IS “Fine synchronism” (DB31-48, DBX98.0) | |

## 4.2 Description of setting data

| | | |
|---|---|---|
| **42300**<br>SD number | **COUPLE_RATIO_1[n].**<br>Speed ratio parameters for synchronous spindle mode [n] | |
| Default setting: 1.0 | Minimum input limit: –1000 | Maximum input limit: 1000 |
| Changes effective after NEW_CONF | Protection level: MMCMD 9220 | Unit: – |
| Data type: DOUBLE | Applies from SW: 2.1 | |
| Meaning: | This setting data determines the speed ratio parameters for the coupling configured in channel-specific MD: COUPLE_AXIS_1[n].<br>The linear correlation between the leading and following spindles is determined by speed ratio $k_Ü$. This ratio is input by two speed ratio parameters in the form of numerator [n=0] and denominator [n=1], allowing the speed ratio to be specified very exactly.<br>$k_{Ü} = \frac{\text{speed ratio parameter numerator}}{\text{speed ratio parameter numerator}} = \frac{\text{\$SC\_COUPLE\_RATIO[0]}}{\text{\$SC\_COUPLE\_RATIO[1]}}$<br>The speed ratio parameters can be altered in the NC part program with language instructions COUPDEF provided this option has not been inhibited with channel-specific MD: COUPLE_IS_WRITE_PROT_1. The parameterized values of SD: $SC_COUPLE_RATIO_1 do not, however, get altered. The calculation of $k_Ü$ is initiated with power ON. | |
| SD irrelevant for ...... | User-defined coupling | |
| Related to .... | SD: $SC_COUPLE_RATIO_1 currently has the same action as a machine data (e.g. active after power ON). The SD data are therefore displayed and input in the same way as channel-specific machine data. | |
| References | Channel-spec. MD: COUPLE_AXIS_1 (definition of synchronous spindle pair) | |

■

# 5 Signal Descriptions

## 5.1 Axis/spindle-specific signals

### 5.1.1 Signals from axis/spindle

| DB31, ...<br>**DBX31.5**<br>Data Block | **Disable synchronization**<br><br>Signal(s) to axis/spindle from PLC (PLC –> NCK) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 6.3 |
| Signal state 1 or signal transition 0 ——> 1 | The synchronization motion for the following spindle is not disabled from the PLC.<br>The position offset is not suppressed and applied as in earlier versions. | |
| Signal state 0 or signal transition 1 ——> 0 | The synchronization motion for the following spindle is disabled from the PLC.<br>A synchronization motion specified via offset programming is suppressed for the following spindle. The following spindle does not execute any additional movement. | |
| Related to .... | IS “Coarse synchronism” (DB31, ... DBX98.1)<br>IS “Fine synchronism” (DB31, ..., DBX98.0) | |

| DB31, ...<br>**DBX84.4**<br>Data Block | **Synchronous mode**<br><br>Signal(s) from axis/spindle to PLC (NCK –> PLC) | |
|---|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 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.<br>The monitoring functions for coarse and fine synchronism are implemented in synchronous operation.<br>Note: The signal is set only for the machine axis which is acting as following spindle (IS “FS active” = 1) | |
| Signal state 0 or signal transition 1 ——> 0 | The spindle is not operated as the following spindle in “synchronous mode”.<br>When the coupling is deactivated (deselection of synchronous operation), the following spindle is switched to “open-loop control mode”. | |
| Related to .... | IS “Coarse synchronism” (DB31-48, DBX98.1)<br>IS “Fine synchronism” (DB31-48, DBX98.0)<br>IS “FS active” (DB31-48, DBX99.1) | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX98.0**<br>Data Block | **Fine synchronism**<br><br>Signal(s) from axis/spindle to PLC (NCK –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The positional deviation or velocity difference between the following spindle and its leading spindle is within the “Fine synchronism” tolerance band (see Subsection 2.1.5). | |
| Signal state 0 or signal transition 1 ––> 0 | The positional deviation or velocity difference between the following spindle and its leading spindle is not within the “Fine synchronism” tolerance band (see Subsection 2.1.5).<br><br>Note: The signal is relevant only for the following spindle in synchronous operation. | |
| Example of an application | 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. | |
| Related to .... | IS “Synchronous operation” (DB31-48, DBX84.4)<br>MD: $MA_COUPLE_POS_TOL_FINE threshold value for fine synchronism or<br>MD: $MA_COUPLE_VELO_TOL_FINE “fine” speed tolerance | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX98.1**<br>Data Block | **Coarse synchronism**<br><br>Signal(s) from axis/spindle to PLC (NCK –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The positional deviation or velocity difference between the following spindle and its leading spindle is within the “Coarse synchronism” tolerance band (see Subsection 2.1.5).<br><br>Note: The signal is relevant only for the following spindle in synchronous operation. | |
| Signal state 0 or signal transition 1 ––> 0 | The positional deviation or velocity difference between the following spindle and its leading spindle is not within the “Coarse synchronism” tolerance band (see Subsection 2.1.5). | |
| Example of an application | 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. | |
| Related to .... | IS “Synchronous operation” (DB31-48, DBX84.4)<br>axis-spec. MD: COUPLE_POS_TOL_COARSE threshold value for “Coarse synchronism” or axis-spec. MD: COUPLE_VELO_TOL_COARSE “coarse” velocity tolerance | |

| | | |
|---|---|---|
| **DB31, ...**<br>**DBX98.2**<br>Data Block | **Actual value coupling**<br><br>Signal(s) from axis/spindle to PLC (NCK –> PLC) | |
| Edge evaluation: no | Signal(s) updated: Cyclic | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The actual-value coupling is active as the coupling type between the leading and following spindles (see channel-specific MD: COUPLING_MODE_1).<br><br>Note: The signal is relevant only for the active following spindle in synchronous operation. | |
| Signal state 0 or signal transition 1 ––> 0 | The setpoint coupling is active as the coupling type between the leading and following spindles (see channel-specific MD: COUPLING_MODE_1). | |
| 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. | |
| Related to .... | IS “Synchronous operation” (DB31-48, DBX84.4)<br>channel-spec. MD: COUPLING_MODE_1 (coupling type in synchr. spindle oper.) | |

| DB31, ...<br>**DBX98.4**<br>Data Block | **Overlaid motion**<br><br>Signal(s) from axis/spindle to PLC (NCK –> PLC) |
|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic — Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The following spindle traverses an additional motional component which is overlaid on the motion from the coupling with the leading spindle.<br><br>Examples of overlaid movement of FS:<br>Activation of synchronous operation with defined angular offset between FS and LS<br>Activation of synchronous operation with LS in rotation<br>Alteration of transmission ratio when synchronous operation is selected<br>Input of a new defined angular offset when synchronous operation is selected<br>Traversal of FS with plus or minus traversing keys or handwheel in JOG when synchronous operation is selected<br><br>As soon as the FS executes an overlaid movement, IS “Fine synchronism” or IS “Coarse synchronism” (depending on threshold value) may be canceled immediately.<br><br>Note: The signal is relevant only for the following spindle in synchronous operation. |
| Signal state 0 or signal transition 1 ––> 0 | The following spindle does not traverse any additional motional component or this motion has been terminated. |
| Related to .... | IS “Synchronous operation” (DB31-48, DBX84.4) |

| DB31, ...<br>**DBX99.0**<br>Data Block | **LS (leading spindle) active**<br><br>Signal(s) from axis/spindle to PLC (NCK –> PLC) |
|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic — Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The machine axis is currently active as the leading spindle.<br><br>Note: The signal is relevant only in synchronous operation. |
| Signal state 0 or signal transition 1 ––> 0 | The machine axis is not currently active as the leading spindle. |
| Related to .... | 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.<br><br>In this case, the leading spindle becomes the new, active following spindle (IS “FS active”). |
| Related to .... | IS “Synchronous operation” (DB31-48, DBX84.4)<br>IS “FS active” (DB31-48, DBX99.1) |

| DB31, ...<br>**DBX99.1**<br>Data Block | **FS (following spindle) active**<br><br>Signal(s) from axis/spindle to PLC (NCK –> PLC) |
|---|---|
| Edge evaluation: no | Signal(s) updated: Cyclic — Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | The machine axis is currently operating as the following spindle.<br><br>The following spindle thus follows the movements of the leading spindle in synchronous operation in accordance with the transmission ratio.<br><br>Note: The signal is relevant only in synchronous operation. |
| Signal state 0 or signal transition 1 ––> 0 | The machine axis is not currently operating as the following spindle. |
| Related to .... | 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. |
| Related to .... | IS “Synchronous operation” (DB31-48, DBX84.4)<br>IS “LS active” (DB31-48, DBX99.0) |

## Notes

# Examples 6

```
                                       ; Leading spindle = master spindle =
                                       ; spindle 1
                                       ; Following spindle = spindle 2
N05   M3 S3000  M2=4 S2=500            ; Leading spindle rotates at 3000 rev/min,
                                       ; FS: –500 rev/min.
N10   COUPDEF (S2, S1, 1, 1, "No", "Dv")          ; Definition of coupling;
                                                  ; can also be configured
...
N70   SPCON                            ; Take leading spindle into position control
                                       ; (setpoint coupling).
N75   SPCON(2)                         ; Take following spindle into position control.
N80   COUPON (S2, S1, 45)              ; Couple on the fly to offset position = 45
                                       ; degrees
...
N200  FA [S2] = 100                    ; Positioning velocity = 100 degrees/min
N205  SPOS[2] = IC(–90)                ; Travel 90 degrees overlaid in the negative
                                       ; direction
N210  WAITC(S2, "Fine")                ; Wait for "fine" synchronism
N212   G1 X.., Y..  F...               ; Processing
...
N215  SPOS[2] = IC(180)                ; Travel 180 degrees overlaid in the positive
                                       ; direction
N220  G4 S50                           ; Dwell time = 50 rotations of
                                       ; master spindle
N225   FA [S2] = 0                     ; Activate configured speed (MD).
N230  SPOS[2] = IC (–7200)             ; 20 rotations with configured speed
                                       ; in the negative direction.
...
N350  COUPOF (S2, S1)                  ; Decouple on the fly, 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
```

■

# Notes

# Data Fields, Lists

7

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| **Channel-specific** | | | |
| 21, ... | 7.1 | NC Start | K1 |
| 21, ... | 7.4 | NC stop axes plus spindle | K1 |
| 21, ... | 24.6 | Dry run feedrate selected | V1 |
| 21, ... | 25.3 | Feedrate override for rapid traverse selected | V1 |
| **Axis/spindle-specific** | | | |
| 31, ... | 1.3 | Axis/spindle disable | A2 |
| 31, ... | 1.4 | Follow up operation | A2 |
| 31, ... | 1.5/1.6 | Position measuring system 1, position measuring system 2 | A2 |
| 31, ... | 2.1 | Controller enable | A2 |
| 31, ... | 2.2 | Spindle RESET | A2 |
| 31, ... | 4.3 | Spindle stop/feed stop | V1 |
| 31, ... | 4.6–4.7 | Traversing keys for JOG | V1 |
| 31, ... | 16.4/16.5 | Re-synchronize spindle 1, re-synchronize spindle 2 | S1 |
| 31, ... | 16.7 | Delete S value | S1 |
| 31, ... | 17.0 | Feedrate override valid | S1 |
| 31, ... | 19 | Spindle override | V1 |
| 31, ... | 31.5 | Disable synchronization | |
| 31, ... | 60.4/60.5 | Referenced/synchronized 1, referenced/synchronized 2 | R1 |
| 31, ... | 84.4 | Synchronous mode | |
| 31, ... | 98.0 | Synchronism fine | |
| 31, ... | 98.1 | Synchronism coarse | |
| 31, ... | 98.2 | Actual value coupling | |
| 31, ... | 98.4 | Superimposed motion | |
| 31, ... | 99.0 | LS/LA active | |
| 31, ... | 99.1 | FS/FA active | |

## 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| **General ($MN_ ...)** | | | |
| 10000 | AXCONF_MACHAX_NAME_TAB | Machine axis name | K2 |
| **Channel-specific ($MC_...)** | | | |
| 21300 | COUPLE_AXIS_1 | Definition of synchronous spindle pair | |
| 21320 | COUPLE_BLOCK_CHANGE_CTRL_1 | Block change behavior in synchronous spindle operation | |
| 21310 | COUPLING_MODE_1 | Type of coupling in synchronous spindle mode | |
| 21330 | COUPLE_RESET_MODE_1 | Coupling abort behavior | |
| 21340 | COUPLE_IS_WRITE_PROT_1 | Coupling parameters are write-protected | |
| 20070 | AXCONF_MACHAX_USED | Machine axis number valid in channel | K2 |
| **Axis/spindle-specific ($MA_...)** | | | |
| 30550 | AXCONF_ASSIGN_MASTER_CHAN | Reset position of channel for axis change | K5 |
| 32200 | POSCTRL_GAIN | Servo gain factor | G2 |
| 32400 | AX_JERK_ENABLE | Axial jerk limitation | B2 |
| 32410 | AX_JERK_TIME | Time constant for axial jerk filter | B2 |
| 32420 | JOG_AND_POS_JERK_ENABLE | Initial setting for axial jerk limitation | B2 |
| 32430 | JOG_AND_POS_MAX_JERK | Axial jerk | B2 |
| 32610 | VELO_FFW_WEIGHT | Feedforward control factor for speed feedforward control | K3 |
| 32620 | FFW_MODE | Feedforward control type | K3 |
| 32650 | AX_INERTIA | Moment of inertia for torque feedforward control | K3 |
| 32800 | EQUIV_CURRCTRL_TIME | Equivalent time constant current control loop for feedforward control | K3 |
| 32810 | EQUIV_SPEEDCTRL_TIME | Equivalent time constant speed control loop for feedforward control | K3 |
| 34080 | REFP_MOVE_DIST | Reference point approach distance | R1 |
| 34090 | REFP_MOVE_DIST_CORR | Home position offset | R1 |
| 34100 | REFP_SET_POS | Reference point value | R1 |
| 35000 | SPIND_ASSIGN_TO_MACHAX | Assignment of spindle to machine axis | S1 |
| 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 | |

## 7.3 Setting data

| Number | Names | Name | Reference |
|---|---|---|---|
| **Axis-specific ($SA_...)** | | | |
| 42300 | COUPLE_RATIO_1 | Transmission parameters for synchronous spindle operation | |

## 7.4 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, “Diagnostics Guide”
and the online help of MMC 101/102/103 systems.

## 7.5 System variable

| System variable | Name | Reference |
|---|---|---|
| $P_COUP_OFFS[following spindle] | Programmed offset of synchronous spindle (SW 6.3 and higher) | PGA 1 |
| $AA_COUP_OFFS[following spindle] | Position offset for synchronous spindle (setpoint) | PGA 1 |
| $VA_COUP_OFFS[following spindle] | Position offset for synchronous spindle (actual value) | PGA 1 |

A more detailed description of system variables can be found in
**References:** /PGA/, “Programming Guide Advanced”.

■

# Notes

# SINUMERIK 840D/840Di/810D
# Description of Functions
# Extended Functions (FB2)

# Memory Configuration (S7)



■

# Notes

# 1 Brief Description

Every CNC requires memory for storing and managing data. This memory can essentially be divided into two areas. One area contains data that doesn't change, such as the software of the CNC. This type of data can be stored on electronic memory chips such as EPROM. The second area contains data stored on the control by the machine manufacturer or user. This data is stored on electronic memory chips such as RAM. The control system enables a RAM area to be set up by the user for various specifications. This description provides information on the areas of RAM that are available to the user and how they can be set up.

**Note**

The SRAM memory currently available is shown in the Program operating area in the program overview (dialog line).

■

## Notes

# Detailed Description 2

## 2.1 General

**Active file system**

The active file system is in the static memory area (SRAM) of the NCK. It holds the system and user data accessed by the NCK during program execution. The active file system is thus not a file system in the true sense, but rather the main memory of the NCK.

The following data are stored in the active file system of the NCK:

- Machine data
- Setting data
- Option data
- Global user data (GUD)
- Too offset / magazine data
- Protection zones
- R variables
- Zero offsets / FRAMEs
- Sag compensations
- Quadrant error compensation
- Leadscrew error compensation

**Passive file system**

The passive file system is located in the static memory area (SRAM) of the NCK. It holds a structure of directories containing files that are permanently stored on the NCK.

Its standard complement of directories is as follows:

| Directory | Contents |
|---|---|
| – _N_MPF_DIR | Standard directory for main programs |
| – _N_SPF_DIR | Subroutines |
| – _N_DEF_DIR | Definition files (*.DEF) of global user data and macros |
| – _N_CST_DIR | Standard cycles |
| – _N_CUS_DIR | User cycles |
| – _N_WKS_DIR | Workpieces |
| – _N_COM_DIR | Standard directory for comments |

When data that can reconfigure the SRAM are changed, all data in the active and passive file systems are lost except for machine data, setting data and option data. It is therefore vital to create a series startup file before such changes are activated.

**SRAM**

The term SRAM refers to the static RAM of the control system that is available to the user for backing up data. SRAM is also referred to in this documentation as backup, battery-backed or static memory.
The SRAM memory currently available is shown in the Program operating area in the program overview (dialog line).

**DRAM**

The term DRAM refers to the dynamic RAM of the control system that is available to the user. The data used in this area are generated exclusively by the control, are only required for a certain length of time and do not, therefore, require backup. DRAM is also referred to in this documentation as an unbuffered or dynamic memory.

## 2.2 Memory organization

**Hardware configuration**

The following table shows the hardware configuration of the available NC CPUs:

| | D-RAM | S-RAM | PCMCIA |
|---|---|---|---|
| NCU 561.x | 32 MB | 4 MB* | 8 MB or more |
| NCU 571.x | 2 x 4 MB | 4 MB* | 8 MB or more |
| NCU 572.x | 32 MB | 4 MB* | 8 MB or more |
| NCU 573.x | 64 MB | 4 MB* | 8 MB or more |

SINUMERIK 840Di

The memory available depends on the hardware configuration of the components used (PCU and MCI board) and the memory available for SINUMERIK 840Di.

| | DRAM Maximum | DRAM for 840Di * | SRAM |
|---|---|---|---|
| PCU 50 | 256 MB | Approx. 16 MB | – |
| MCI board | – | – | 1 MB |

*) DRAM component (main memory) occupied by SINUMERIK 840Di and thus no longer available for Windows NT.

**SRAM**

SRAM that is available to the user. It can be configured by means of the machine data described in this Description of Functions.

- NCU 571.x 256 KB/ 1.5 MB*
- NCU 572.x 256 KB/ 1.5 MB*
- NCU 573.x 256 KB/ 1.5 MB*

SINUMERIK 840Di

- MCI board approx. 0.5 MB

In the machine data 18230: MM_USER_MEM_BUFFERED. For NCUs with 2 different memory capacities, the standard entry takes the smallest value into account. If the 2.0MB version is used, then you have to set MD 18230 explicitly to 1900. (Although the gross value is 2000, it is necessary to make a deduction for internal use).

**DRAM**

The total amount of DRAM available to the user is displayed in MD 18210: MM_USER_MEM_DYNAMIC (dynamic user memory in DRAM). The value is system-dependent and may vary slightly with different software versions.

The memory areas containing the individual data groups, e.g. global user data, channel-specific user data, axis-specific user data, etc., are arranged contiguously in SRAM and DRAM. The size of a data area in use can be configured in the appropriate machine data. The order in which the data areas are arranged is permanently defined by the CNC.

**Alteration of memory areas**

It is evident from the memory organization described above that any changes to the memory areas must affect the data stored. Every time the system powers up, the CNC compares the current requirement for memory with the existing memory space on the basis of machine data for individual data areas. If the comparison establishes that one or more modified data areas require reallocation of the data areas, the memory is reorganized.

**Loss of data during reconfiguration**

Reconfiguring the memory always causes total loss of the entire memory contents (i.e. contents of active and passive file system).

All data stored in the active and passive file systems will also be lost if the total SRAM and/or DRAM memory requirement specified via machine data exceeds the amount of available memory. For this reason, it is absolutely essential to save the data in a series startup file before modifying the memory configuration.

The amount of memory space available prior to a reconfiguration is displayed in machine data

- MD18060: INFO_FREE_MEM_DYNAMIC (free static memory)
- MD18050: INFO_FREE_MEM_DYNAMIC (free dynamic memory).

displayed.

**Memory-configuring machine data**

The following list shows some of the machine data which affect the memory configuration:

- System-specific memory settings
- Channel-specific memory settings
- Axis-specific memory settings.

The modification of machine data for the SRAM belonging to these groups always leads to the loss of data in the backup memory. The name of these machine data begins with MM_ (e.g. MM_NUM_TOA_MODULES).

The number of active channels also affects the memory configuration. If the number of channels is altered, these channels are set up according to the default settings for the channel-specific memory areas when the system is powered up. Since these areas are also in the SRAM, changing the number of channels also leads to a loss of data in the backup user memory.

After a new value has been entered in a machine data which reallocates the memory area of the SRAM (see Section 2.4), message 4400 “MD alteration will cause reorganization of buffer (data loss!)” is output. This warning indicates that a machine data has been changed, which causes the backup memory to be reorganized when the NCK is powered up, resulting in a loss of all user data stored there. If the memory is to be reorganized and the control contains data which have not been backed up, these should be saved before the next NCK power-up.

---

**Note**

The reorganization can be avoided by changing the modified value back to the original value at the time of the last power-up.

---

Only in exceptional cases does an MD change not cause reorganization of the memory!

In the case of MD 18350: MM_USER_FILE_MM_MINIMUM (minimum parts program memory), the memory reorganization is only performed if the remaining RAM is too small.

Loading the memory-configuring standard machine data on the next system power-up through setting system-specific MD 11200: INIT_MD (load standard machine data on next power-up) to 2 causes the backed up user data to be lost if the memory areas are not organized according to the memory default settings before the system powers up.

## 2.3 Memory configuration alarms

A modification to the memory allocation that is incorrect or requires memory reorganization causes the output of an alarm message after CNC system power-up. The causes of the faults and the response of the CNC can be summarized as follows:

**Alarm 6000**

The user memory (static or dynamic) cannot be reallocated because the total memory area available (static or dynamic) is less than the total number of memory areas set by machine data. In this case, all machine data for configuring memory are deleted and assigned their default values. NC machining is no longer possible. This situation is indicated by alarm 6000 "Memory allocation with standard machine data". It is not possible to pinpoint a particular machine data as the cause of the error in memory allocation. However, it is possible to find the error by altering the machine data for the memory settings one by one. The alarm can be canceled with RESET. Machining is possible only when the user data have been loaded.

**Alarm 6010**

After cycle programs, macro definitions or definitions of global user data have been incorporated, alarm 6010 "Channel [name 1] data block [name 2] has not been or is only partially created, error number [identifier]" is output in response to an error. Either the machine data for the corresponding memory areas have been incorrectly configured or the files contain an error. As an example, three files for macro definitions _N_SMAC_DEF, _N_MMAC_DEF and _N_UMAC_DEF contain a total of 30 macro definitions, but the setting in MD 18160: MM_NUM_USER_MACROS (number of macros) restricts the number of macros to 10.

The identifier [name 1] indicates the name of the channel where the error has occurred. The identifier [name 2] indicates the name of the file with the error. The error number is coded as follows with respect to the cause:

| Error number | Explanation |
|---|---|
| 1 | No memory available |
| 2 | Maximum no. of symbols exceeded |
| 3 | Index 1 outside permissible value range |
| 4 | Name already exists on channel |
| 5 | Name already exists on NCK |
| >100 000 | Unrecoverable system error |

If the error number output is between 1 and 5, the user can eliminate the error himself. In cases where the error number is >100000, the error is an unrecoverable system error.

Machining is possible when the machine data or files have been corrected, or the changes have been canceled and the system rebooted.

**Alarm 6020**

The SRAM has been reorganized after a modification to the static memory allocation. All stored data, with the exception of the machine data, have been lost. Alarm 6020 “Machine data altered – memory reallocated” indicates this situation. The SRAM is reallocated when the number of channels on the CNC or the system, channel or axis-specific memory settings for the static memory are altered. The alarm can be canceled with RESET. Machining can resume when the user data are loaded.

**Alarm 6030**

The memory area set in MD 18210: MM_USER_MEM_DYNAMIC (user memory in DRAM), MD 18220: MM_USER_MEM_DPR (user memory in dual-port RAM) or MD 18230: MM_USER_MEM_BUFFERED (user memory in SRAM) is larger than the physical memory actually available. In this case, the CNC enters the available memory in the corresponding machine data and displays it with alarm 6030 “User memory limit has been adapted”. In this case, no user data are lost. This alarm message can be cleared by a RESET.
Alarm 6000 “Memory allocation with standard machine data” may arise, however, if further machine data were used for the memory allocation assuming that the excessively large data is correct and memory has been allocated over and above the area actually available.

## 2.4 Memory allocation in SRAM and DRAM

Since in normal practice the SRAM and DRAM memory is only allocated as part of the start-up process, we would recommend the following procedure for allocating memory taking the SRAM as an example:

- Load standard machine data.
- MD 18230: MM_USER_MEM_BUFFERED (user memory in SRAM) is set to a high value (3000). The NCK is then powered up. Alarm 6030 "User memory limit has been adapted" is output and the maximum amount of memory available to the user entered in MD 18230: MM_USER_MEM_BUFFERED.
  All other memory-configuring machine data are set to their defaults.
- Activate the number of channels and axes required, for further details see
  **References:** / IAD / Installation & Start-Up Guide SINUMERIK 840D
  / HBI / SINUMERIK 840Di Manual
- The static memory still available is displayed in MD 18060: INFO_FREE_MEM_STATIC (display data of free static memory).
- If the memory default settings do not allocate the memory satisfactorily, then the memory areas can now be reconfigured (increase/decrease individual or several areas via machine data) to adapt the memory provided to the requirements on the machine tool.
  Check: Which memory areas require more memory space?
  Which memory areas are less important for the application in question?
- After the appropriate machine data for the selected memory areas have been set to define memory requirements, the NCK is reset in order to reorganize the memory.

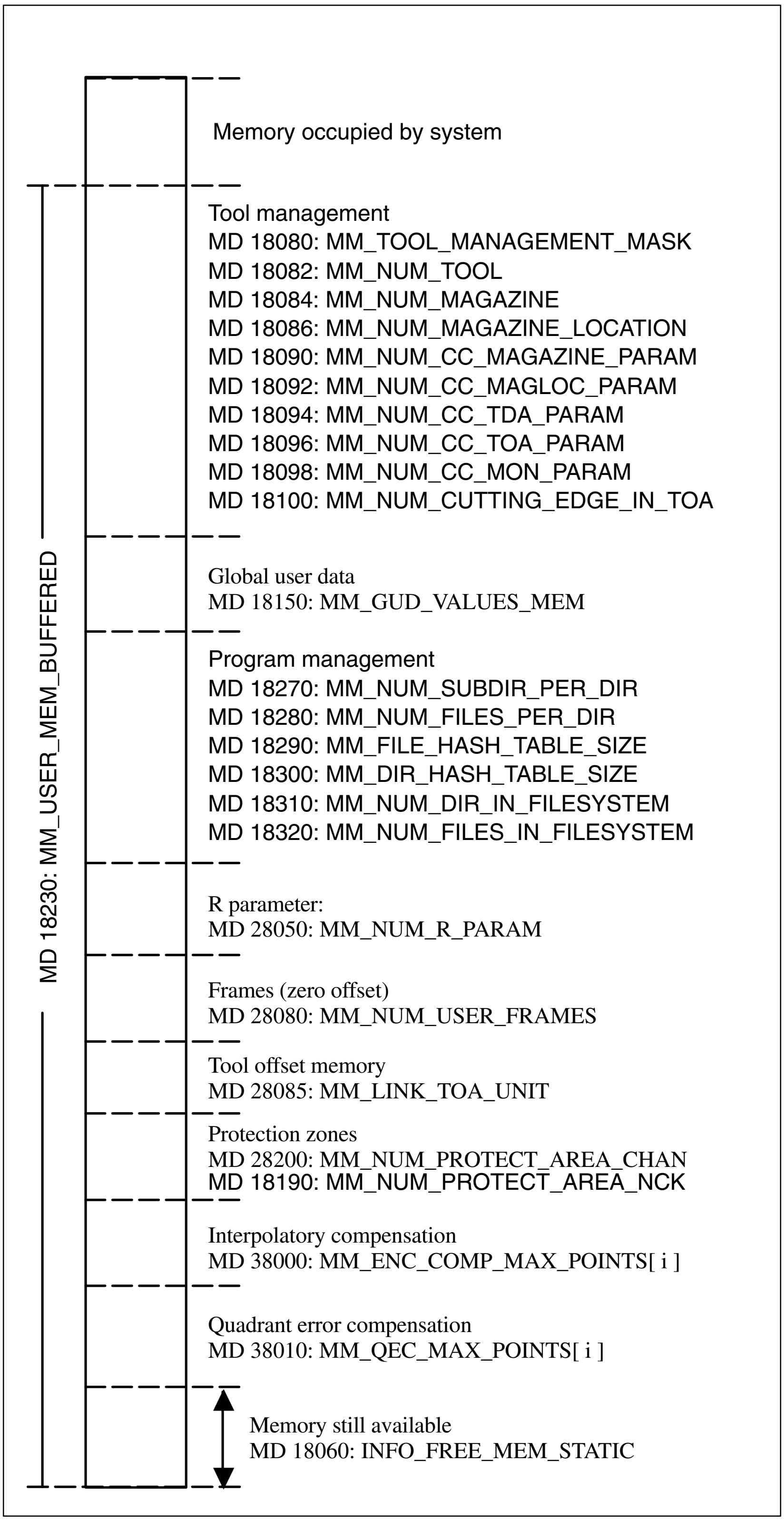



Fig. 2-1 Allocation of static RAM (SRAM)

### 2.4.1 Memory allocation SRAM

Table 2-1 Allocation of memory space in SRAM

| Machine data | Memory requirements | Remarks |
|---|---|---|
| | **Tool management** | |
| MD 18080: MM_TOOL_MANAGEMENT_MASK | | See detailed description of MD in Chapter 4. |
| MD 18082: MM_NUM_TOOL | 84 bytes per tool | |
| MD 18084: MM_NUM_MAGAZINE | 332 bytes per magazine | |
| MD 18086: MM_NUM_MAGAZINE_LOCATION | 64 bytes per magazine location | |
| MD 18090: MM_NUM_CC_MAGAZINE_PARAM | Input x no. of magazines x 4 bytes | Corresponds to MD 18084: MM_NUM_MAGAZINE |
| MD 18092: MM_NUM_CC_MAGLOC_PARAM | Input x no. of magazines x 4 bytes | Corresponds to MD 18084: MM_NUM_MAGAZINE |
| MD 18094: MM_NUM_CC_TDA_PARAM | Input x no. of tools x 4 bytes | Corresponds to MD 18082: MM_NUM_TOOL |
| MD 18096: MM_NUM_CC_TOA_PARAM | Input x no. of TOs x 8 bytes | Corresponds to MD 18100: MM_NUM_CUTTING_EDGES_ IN_TOA |
| MD 18098: MM_NUM_CC_MON_PARAM | Input x no. of TOs x 4 bytes | Corresponds to MD 18100: MM_NUM_CUTTING_EDGES_ IN_TOA |
| MD 18100: MM_NUM_CUTTING_EDGES_IN_ TOA | Without active monitor: 250 bytes per tool edge<br><br>With active monitor: Additional 48 bytes per tool edge | |
| | **Global user data** | |
| MD 18118: MM_NUM_GUD_MODULES | | See following example |
| MD 18120: MM_NUM_GUD_NAMES_NCK | 80 bytes per NCK name | See following example |
| MD 18130: MM_NUM_GUD_NAMES_CHAN | 80 bytes per channel name | See following example |
| MD 18140: MM_NUM_GUD_NAMES_AXIS | 80 bytes per axis name | |
| MD 18150: MM_GUD_VALUES_MEM | | See following example |

**Example of GUD**

An example of how the calculate the memory requirements of global user data is given below.

Supplementary conditions

- Machine with two channels.
- The following GUD modules are defined:
  UGUD User-specific
  SGUD Siemens-specific
  MGUD Machine manufacturer-specific
  GUD7 (Contour table stock removal cycle, required for CYCLE95, cycle version 3.4 and higher)
- NCK-specific and channel-specific variables are defined.

NCK variables

```
2        REAL values    -> 2 x 8 bytes
= 16 bytes
1        BOOL values    -> 1 x 1 byte
= 1 byte
               Total 1  = 17 bytes
3 = Total NCK (no. of values)
```

CHAN variables

```
2        BOOL values    -> 2 x 1 bytes
= 2 bytes
1        INT values     -> 1 x 4 bytes
= 4 bytes
               Total 2  = 6 bytes
3 = Total CHAN (no. of values)

6 bytes (total 2) x 2 (no. of channels) = 12 bytes (total 3)
```

Calculation of memory required

- MD 18120: MM_NUM_GUD_NAMES_NCK = 3 (total NCK)
  Memory space for management of NCK names
  => 3 x 80 bytes = **240 bytes**
- MD 18130: MM_NUM_GUD_NAMES_CHAN = 3 (total CHAN)
  Memory space for management of CHAN names
  => 3 x 80 bytes = **240 bytes**
- Number of max. defined GUD modules = 7 (GUD7)
  Memory space for management of GUD modules
  => 7 x 120 bytes = **840 bytes**
- Memory requirements for variables
  Total 1 + total 3 =
  17 bytes + 12 bytes = **29 bytes**, rounded up to whole KB gives:
  MD 18150: MM_GUD_VALUES_MEM = 1
- Total memory space required for GUD is calculated as:

| | |
|---|---|
| Memory space for management of NCK names | = 240 bytes |
| Memory space for management of CHAN names | = 240 bytes |
| Memory space for management of GUD modules | = 840 bytes |
| Memory space required for variables | = 1024 bytes |
| **Total** | = **2344 bytes** |

| Machine data | Memory requirements | Remarks |
|---|---|---|
| | **Program management** | |
| MD 18270: MM_NUM_SUBDIR_PER_DIR | 40 bytes per subdirectory | |
| MD 18280: MM_NUM_FILES_PER_DIR | 40 bytes per file | |
| MD 18290: MM_FILE_HASH_TABLE_SIZE | | Assigned internally by the system and must not be altered by user. |
| MD 18300: MM_DIR_HASH_TABLE_SIZE | | Assigned internally by the system and must not be altered by user. |
| MD 18310: MM_NUM_DIR_IN_FILESYSTEM | | See detailed description of MD in Chapter 4. |
| MD 18320: MM_NUM_FILES_IN_FILESYSTEM | 320 bytes per file | |
| | **R variables** | |
| MD 28050: MM_NUM_R_PARAM | 8 bytes per R parameter | |
| | **Frames (zero offsets)** | |
| MD 28080: MM_NUM_USER_FRAMES | 232 bytes per frame<br><br>An additional 120 bytes are required once for management purposes. | |
| | **Tool offset memory** | |
| MD 28085: MM_LINK_TOA_UNIT | 500 bytes per unit | |
| | **Protection zones** | |
| MD 28200: MM_NUM_PROTECT_AREA_CHAN | 400 bytes for each defined block | |
| MD 18190: MM_NUM_PROTECT_AREA_NCK | 400 bytes for each defined area | |
| | **Compensation** | |
| MD 18340: MM_NUM_CEC_NAMES | 1 KB permanently allocated | |
| MD 18342: MM_CEC_MAX_POINTS | 8 bytes per compensation point<br><br>An additional 2 bytes are required once for management purposes. | |
| MD 38000: MM_ENC_COMP_MAX_POINTS | 8 bytes per compensation point<br><br>An additional 2 bytes are required once for management purposes. | |
| MD 38010: MM_QEC_MAX_POINTS | 4 bytes per compensation point<br><br>An additional 2 bytes are required once for management purposes. | |

### 2.4.2 Memory allocation DRAM

- The dynamic memory still available is displayed in MD 18050: INFO_FREE_MEM_DYNAMIC (display data of available dynamic memory).
- If the memory default settings do not allocate the memory satisfactorily, then the memory areas can now be reconfigured (increase/decrease individual or several areas) to adapt the memory provided to the requirements on the machine tool.
  Check: Which memory areas require more memory space?
  Which memory areas are less important for the application in question?
- After the appropriate machine data for the selected memory areas have been set to define memory requirements, the NCK is reset in order to reorganize the memory.

---

**Note**

If more dynamic memory is demanded than the amount actually available, the SRAM(!) is also automatically deleted during the next power-up and all machine data for the memory configuration reset to their default values.

---

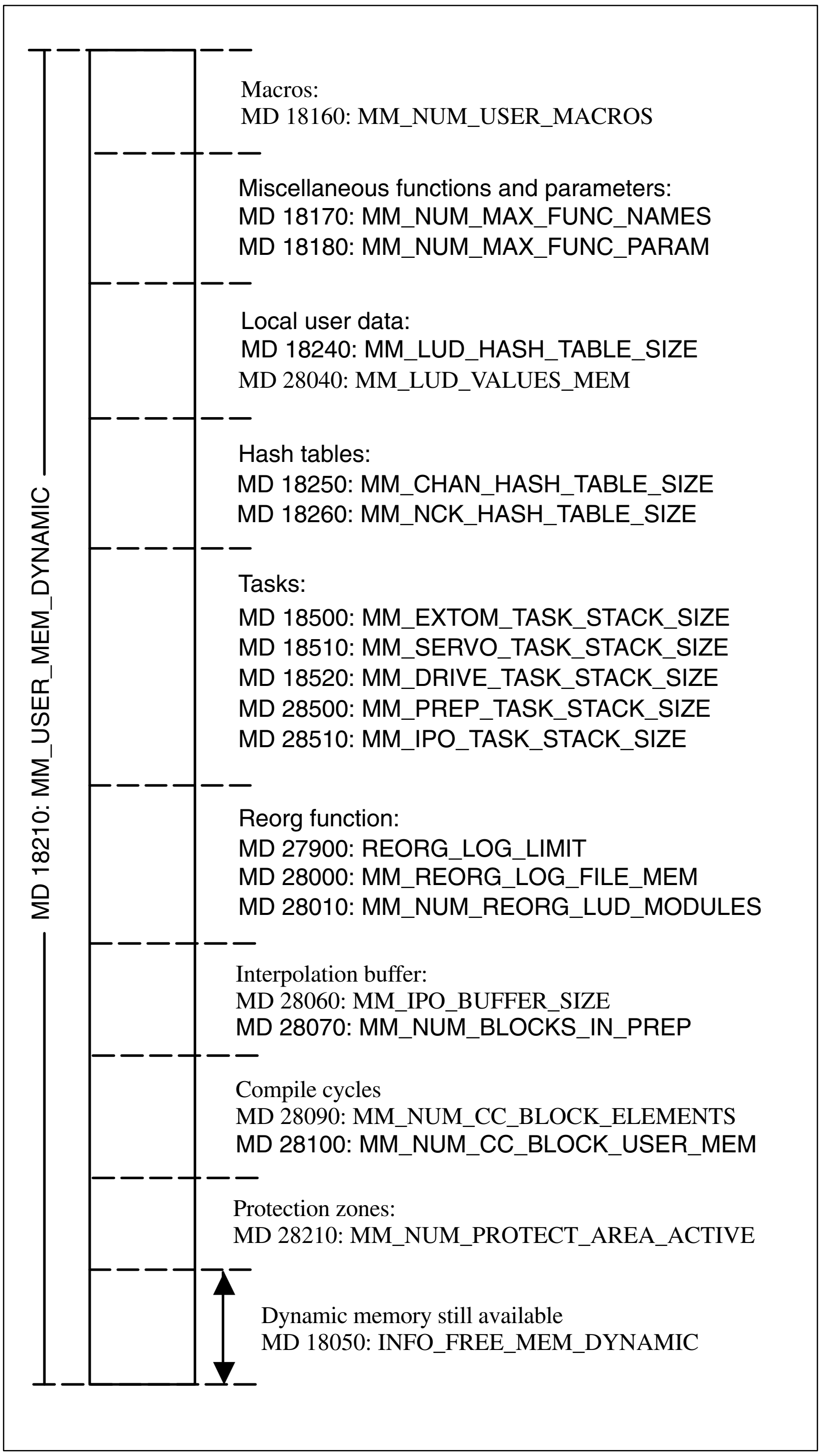



Fig. 2-2 Allocation of memory space in dynamic RAM (DRAM)

| Machine data | Memory requirements | Remarks |
|---|---|---|
| | **Macros** | |
| MD 18160:<br>MM_NUM_USER_MACROS | 375 bytes per macro | |
| | **Miscellaneous functions and their additional parameters** | |
| MD 18170:<br>MM_NUM_MAX_FUNC_NAMES | 150 bytes per miscellaneous function | |
| MD 18180:<br>MM_NUM_MAX_FUNC_PARAM | 72 bytes per parameter | The entered value is the total of all miscellaneous function parameters required. |
| | **Local user data** | |
| MD 18240:<br>MM_LUD_HASH_TABLE_SIZE | | Assigned internally by the system and must not be altered by user. |
| MD 18242:<br>MM_MAX_SIZE_OF_LUD_VALUE | Block size depends on variable used<br><br>Data type Memory requirement<br>REAL 8 bytes<br>INT 4 bytes<br>BOOL 1 byte<br>CHAR 1 byte<br>STRING 1 byte per character, 100 characters permitted per string<br>AXIS 4 byte<br>FRAME 400 bytes | The machine data must be set for the variable that requires the most memory space. However, the machine data must not be set higher than this variable or an alarm message is output.<br><br>See following example. |
| MD 28010:<br>MM_NUM_REORG_LUD_MODULES | | Assigned internally by the system and must not be altered by user. |
| MD 28020:<br>MM_NUM_LUD_NAMES_TOTAL | 150 bytes per LUD name | See following example. |
| MD 28040:<br>MM_LUD_VALUES_MEM | Total memory space required for LUD | See following example. |

**Example of local user data**

Local user data defined in the parts programs are stored in the DRAM while the program in which they are defined is being executed.
An example of how to calculate the memory requirements of local user data is given below.

Parameters of example

The following variables must be used:
1 REAL value -> 1 x 8 bytes
= 8 bytes
2 BOOL values -> 2 x 1 byte
= 2 bytes
Total 1 = 10 bytes
3 = Total A (no. of values)

Calculation of memory required

- Memory required for variables
  Total 1 = **10 bytes**
- MD 18242: MM_MAX_SIZE_OF_LUD_VALUES = 8 bytes
  The machine data must be set according to the variable that requires the most memory space. In this above example, this is the REAL value with 8 bytes.
- MD 28020: MM_NUM_LUD_NAMES_TOTAL = 3 (total A)
  Memory space for management of LUD NAMES
  => 3 x 150 bytes = **450 bytes**
- MD 28040: MM_LUD_VALUES_MEM

  Total memory space required for LUD is calculated as:

| | | |
|---|---|---|
| Memory space for management of LUD NAMES | = | 450 bytes |
| Memory space required for variables | = | 10 bytes |
| **Total** | = | **456 bytes** |

  The calculated sum must be rounded in KB and entered in
  MD 28040: MM_LUD_VALUES_MEM (in this case, 1KB).
  The memory provided by this setting is allocated in blocks of 8 bytes each in size (according to MD 18242).
  If, for example, 1 REAL value (8 bytes) and 1 BOOL value (1 byte) are used in a program, then 2 blocks of memory of 8 bytes each are allocated.

**LUD defined in parts programs**

Local user data defined in the parts programs are stored in the DRAM while the program in which they are defined is being executed.
During this period, it is possible to view the assigned values under the softkey PARAMETER.

Definition of variables in PP

| | |
|---|---|
| DEF INT LUD_VARIABLE1 | Integer variable with the name VARIABLE1 |
| DEF REAL LUD_VARIABLE2 | REAL variable with the name VARIABLE2 |
| DEF REAL LUD_PAUL[19] | Field with 20 REAL variables PAUL[0] – PAUL[19] |

Memory management

The system automatically controls the allocation of memory blocks.

- Reservation of a memory block when a parts program containing the LUD definition is processed.
- Reservation of further blocks if the memory provided for the number of variables is not sufficient.
- Release of memory space if LUD are no longer required (at end of program).

Variants of variable definition

When a large number of variables is required, e.g. 20 REAL variables, it is possible to save memory space by defining an ARRAY (field) at the beginning of a parts program rather than defining each variable individually.

**Example:**
**Case1**
DEF REAL LUD_PAUL[19]
This field with 20 LUD variables PAUL[0] – PAUL[19] requires the following memory space:

| | | | |
|---|---|---|---|
| MD 28080 = 1 | => | 1 x 150 bytes = | 150 bytes |
| Memory for 20 variables | => | 20 x 8 bytes = | 160 bytes |

Total memory required by 20 variables = 310 bytes

**Case2**
Individual definition of 20 variables: PAUL0, PAUL1 – PAUL19

| | | | |
|---|---|---|---|
| MD 28080 = 20 | => | 20 x 150 bytes = | 3000 bytes |
| Memory for 20 variables | => | 20 x 8 bytes = | 160 bytes |

Total memory required by 20 variables = 3160 Byte

---

**Note**

This alternative method of variables definition can also be applied to GUD variables.

See MD: MM_MAX_SIZE_OF_LUD_VALUES for LUD variables.

---

| Machine data | Memory requirements | Remarks |
|---|---|---|
| **Hash tables** | | |
| MD 18250:<br>MM_CHAN_HASH_TABLE_SIZE | Input x no. of channel-spec. names x 68 bytes | Assigned internally by the system and must not be altered by user. |
| MD 18260:<br>MM_NCK_HASH_TABLE_SIZE | Input x no. of NCK-spec. names x 68 bytes | Assigned internally by the system and must not be altered by user. |
| **Tasks** | | |
| MD 18500:<br>MM_EXTCOM_TASK_STACK_SIZE | Input x 1 KB | |
| MD 18510:<br>MM_SERVO_TASK_STACK_SIZE | Input x 1 KB | |
| MD 18520:<br>MM_DRIVE_TASK_STACK_SIZE | Input x 1 KB | |
| MD 28500:<br>MM_PREP_TASK_STACK_SIZE | Input x 1 KB | |
| MD 28510:<br>MM_IPO_TASK_STACK_SIZE | Input x 1 KB | |
| **Reorg function** | | |
| MD 27900:<br>REORG_LOG_LIMIT | Input x 1 KB | Assigned internally by the system and must not be altered by user. |
| MD 28000:<br>MM_REORG_LOG_FILE_MEM | Input x 1 KB | Assigned internally by the system and must not be altered by user. |
| MD 28010:<br>MM_NUM_REORG_LUD_MODULES | Input x 1 KB | Assigned internally by the system and must not be altered by user. |
| **Interpolation buffer** | | |
| MD 28060:<br>MM_IPO_BUFFER_SIZE | 10 KB for each NC block in IPO buffer | |
| MD 28070:<br>MM_NUM_BLOCKS_IN_PREP | 10 KB for each NC block for preparation | |
| **Compile cycles** | | |
| MD 28090:<br>MM_NUM_CC_BLOCK_<br>ELEMENTES | 1.2 KB per block element | |
| MD 28100:<br>MM_NUM_CC_BLOCKS_USER_<br>MEM | Input / 128 bytes = no. of blocks | The entered value should be a multiple of 128 since the memory is enabled in a grid of 128-byte blocks. |
| **Active protection zones** | | |
| MD 28210:<br>MM_NUM_PROTECTED_AREA_<br>ACTIVE | | The value entered should be determined by the total of the settings in MD 18190 and MD 28200.<br>MD 18190 = 2<br>MD 28200 = 2<br>=> MD 28210 = 4 |

## 2.5 Memory requirements calculation

---

**Note**

The memory required depends on the software version and type of NC control. The values specified in the table below for the change in memory requirements based on changes in machine data are intended as **guide values** for SW 4 and NCU 572. The defaults and machine data limits for other software versions or other NC controls can be found in:
**References:** /LIS/, Lists (for relevant software version)

---

**Overview**

The tables are arranged in the following order:

- DRAM
  - General machine data
  - Channelspecific machine data
  - Axis-specific machine data
- SRAM
  - General machine data
  - Channelspecific machine data
  - Axis-specific machine data

### Table entries

1. **MD no.**
   Number of the machine data. The associated identifier can be looked up in /LIS/.
2. **Meaning**
   Meaning of the machine data.
   New line:
   **GD:** Basic DRAM overhead, **GS:** Basic SRAM overhead
   (This overhead is produced when the function controlled by the MD is used. Values are only specified for MD which are not directly proportional to the value specified in column 3 or which cannot be calculated.)
3. **Default value** (def)
   Value set when the software is supplied.
4. **Increase def. by 1, extra req. (bytes)**
   Specifies the number of bytes by which the memory requirement changes if the default value is increased by 1.
   The basic overheads for GD and GS specified in column 2 are included in the changes shown.
5. **Increase def. by further (n)**
   Specifies by how many additional units the value of the machine data was increased in the capacity calculation. The increased memory allocation is specified in column 6.
6. **Extra requirement for n, (bytes)**
   Specifies how much additional memory is required if the machine data is increased by the value specified in column 5.
   The basic overheads for GD and GS specified in column 2 are included in the changes shown.
7. **SRAM also affected**
   **DRAM also affected**
   An x appears in the column if the other type of memory is also occupied proportionally.

---

**Note**

The actual dependencies between machine data and required memory are complex. Some MD initiate further functions which also use memory. The relationship between the amount of memory used and the number in the MD is not always linear. The tables below therefore only provide an approximate indication of where memory can be reduced or increased in order to achieve the desired configuration. The information applies both to increasing and reducing the values specified in the machine data.

---

### 2.5.1 DRAM memory requirements

Table 2-2 General machine data, DRAM

| MD number | Meaning | Default value | Increase def. by 1, extra req. (bytes) | Increase def. by further (n) | Extra req. for n (bytes) | SRAM also affected |
|---|---|---|---|---|---|---|
| 10010 | Channels | 1 | 1134608 | | See SRAM | x |
| 10134 | Number of MMC communication partners | 6 | 28236 | | | |
| 18082 | Number of tools | 30 | 120 | 10 | 1244 | x |
| 18088 | Number of toolholders **GD:** 588, **GS:** 1293 | 0 | 588 | n | See SRAM | x |
| 18120 | Number of global user variables | 10 | 84 | 10 | 828 | x |
| 18130 | No. of channel-specific user variables | 40 | 84 | 10 | 828 | x |
| 18140 | No. of axis-specific user variables | 0 | 84 | 10 | 828 | x |
| 18160 | No. of macros | 10 | 680 | 10 | 6864 | |
| 18170 | No. of miscellaneous functions (cycles) | 40 | 120 | 10 | 1272 | |
| 18180 | No. of additional parameters for cycles | 300 | 60 | 10 | 612 | |
| 18190 | Number of files for machine-related protection zones **GD:** 504, **GS**: 1062 | 0 | 504 | n | See SRAM | x |
| 18210 | Dynamic user memory | 3370 | 1024 | 10 | 10240 | |
| 18280 | No. of files per directory | 100 | 76 | | | |
| 18342 | Max. number of interpolation points for beam sag compensation **GD:** 380, **GS:** 1680 | 0 | 380 | 10 | See SRAM | x |
| | | | | | | |

Table 2-3 Channel-specific machine data, DRAM

| MD number | Meaning | Default value | Increase def. by 1, extra req. (bytes) | Increase def. by further (n) | Extra req. for n (bytes) | SRAM also affected |
|---|---|---|---|---|---|---|
| 28000 | Memory size for REORG | 10 | 1084 | 10 | 10636 | |
| 28020 | Number of local user variables | 200 | 160 | 10 | 1688 | |
| 28040 | Memory capacity for local user variables | 8 | 1024 | 10 | 10260 | |
| 28060 | No. of NC blocks in IPO buffer | 10 | 15452 | | | |
| 28070 | No. of blocks for block preparation | 36 | 15576 | | | |
| 28080 | No. of settable frames | 5 | 76 | 10 | 784 | x |
| 28085 | Assignment of TOA unit to a channel | 1,2,3... | 84 | | See SRAM | x |
| 28090 | No. of block elements for compile cycles | 0 | 924 | | | |

Table 2-3 Channel-specific machine data, DRAM

| MD number | Meaning | Default value | Increase def. by 1, extra req. (bytes) | Increase def. by further (n) | Extra req. for n (bytes) | SRAM also affected |
|---|---|---|---|---|---|---|
| 28100 | Capacity of block memory for compile cycles | 256 | 1056 | | | |
| 28150 | Number of elements for writing PLC variables | 0 | 56 | | | |
| 28200 | Number of files for channel-spec. protection zones<br>**GD:** 504, **GS**: 1062 | 0 | 504 | | See SRAM | x |
| 28210 | Number of simultaneously active protection zones | 0 | ~18000 | 10 | 174852 | |
| 28250 | Number of elements for expressions in synchronized actions | 159 | 104 | | | |
| 28252 | Number of elements for FCTDEF definitions | 3 | 32 | | | |

Table 2-4 Axis-specific machine data, DRAM

| MD number | Meaning | Default value | Increase def. by 1, extra req. (bytes) | Increase def. by further (n) | Extra req. for n (bytes) | SRAM also affected |
|---|---|---|---|---|---|---|
| 38000 | No. of interpolation points for interpolation compensation<br>**GD:** 212, **GS:** 976 | 0 | | 10 | 212 | x |
| 38010 | Number of values for quadrant error compensation<br>**GD:** 548, **GS:** 1932 | 0 | | 10 | 548 | x |
| | | | | | | |

## 2.5.2 SRAM memory requirements

Table 2-5 General machine data, SRAM

| MD number | Meaning | Default value | Increase def. by 1, extra req. (bytes) | Increase def. by further (n) | Extra req. for n (bytes) | DRAM also affected |
|---|---|---|---|---|---|---|
| 10010 | Channels | 1 | 10032 | | | x |
| 18082 | Number of tools | 30 | 80 | 10 | 812 | x |
| 18084 | Number of magazines | 3 | 244 | 10 | 2416 | |
| 18086 | Number of magazine locations<br>**GD:** 0, **GS:** 6 | 30 | 244 | 31 | 7612 | |
| 18088 | Number of toolholders<br>**GD:** 588, **GS:** 1293 | 0 | 1408 | 10 | 1152 | x |
| 18090 | Quantity of magazine data for compile cycles<br>**GD**: 0, **GS:** 36 | 0 | 40 | 10 | 32 | |
| 18092 | Quantity of magazine location data for compile cycles<br>**GD**: 0, **GS:** 36 | 0 | 40 | 10 | 32 | |
| 18094 | Number of tool-spec. data per tool<br>**GD:** 0, **GS:** 31 | 0 | 40 | 9 | 68 | |
| 18096 | Quantity of data per tool edge<br>**GD:** 0, **GS:** 31 | 0 | 40 | 9 | 68 | |
| 18098 | Quantity of monitoring data per tool edge<br>**GD**: 0, **GS:** 36 | 0 | 40 | 9 | 32 | |
| 18100 | Tool offsets per TOA module<br>**GD:** 0, **GS:** 13 | 30 | 244 | 10 | 2408 | |
| 18102 | Type of D number programming | 0 | –2344<br>(reduced requirement for 1: direct D no. prog.) | | | |
| 18118 | Number of GUD files in active file system | 7 | 628 | | | |
| 18120 | Number of global user variables | 10 | 120 | 10 | 1200 | x |
| 18130 | No. of channel-spec. user variables | 40 | 120 | 10 | 1200 | x |
| 18140 | No. of axis-spec. user variables | 0 | 120 | 10 | 1200 | x |
| 18150 | Memory location for user variables | 12 | 1056 | 10 | 10548 | |
| 18190 | Number of files for machine-related protection zones<br>**GD:** 504, **GS**: 1062 | 0 | 1464 | 5 | 2012 | x |
| 18230 | User memory in SRAM | 1900 | 1024 | 10 | 10232 | |
| 18310 | No. of directories in passive file system | 30 | 1236 | | | |
| 18320 | No. of files in passive file system | 100 | 344 | | | |
| 18342 | Number of interpolation points for beam sag compensation<br>**GD:** 380, **GS:** 1680 | 0 | | 10 | 1748 | x |

Table 2-5 General machine data, SRAM

| MD number | Meaning | Default value | Increase def. by 1, extra req. (bytes) | Increase def. by further (n) | Extra req. for n (bytes) | DRAM also affected |
|---|---|---|---|---|---|---|
| 18400 | Number of curve tables<br>**GD:** 0, **GS:** 4 | 0 | 104 | 1 | 100 | |
| 18402 | Number of curve segments<br>**GD:** 0, **GS:** 4 | 0 | 128 | 1 | 124 | |
| 18404 | Number of curve table polynomials<br>**GD:** 0, **GS:** 4 | 0 | 60 | 1 | 56 | |

Table 2-6 Channel-specific machine data, SRAM

| MD number | Meaning | Default value | Increase def. by 1, extra req. (bytes) | Increase def. by further (n) | Extra req. for n (bytes) | DRAM also affected |
|---|---|---|---|---|---|---|
| 28050 | Number of channel-specific R parameters | 100 | 8 | 10 | 64 | |
| 28080 | No. of settable frames | 5 | 428 | 10 | 4220 | x |
| 28085 | Assignment of TOA unit to a channel | 1,2,3 ... | 2124 | | | x |
| 28200 | Number of files for channel-specific protection zones<br>**GD:** 504, **GS**: 1062 | 0 | 1468 | 5 | 2008 | x |
| | | | | | | |

Table 2-7 Axis-specific machine data, SRAM

| MD number | Meaning | Default value | Increase def. by 1, (bytes) | Increase def. by further (n) | Extra req. (bytes) | DRAM also affected |
|---|---|---|---|---|---|---|
| 38000 | No. of interpolation points for interpolation compensation<br>**GD:** 212, **GS:** 976 | 0 | | 10 | 1040 | x |
| 38010 | Number of values for quadrant error compensation<br>**GD:** 548, **GS:** 1932 | 0 | | 10 | 1996 | x |
| | | | | | | |

■

# Supplementary Conditions

3

None

# Data Descriptions (MD, SD)

4

## 4.1 General machine data

| | | | | | |
|---|---|---|---|---|---|
| **18050**<br>MD number | **INFO_FREE_MEM_DYNAMIC**<br>Display data of free dynamic memory | | | | |
| Default setting: 1048576 | | Minimum input limit: – | | Maximum input limit: – | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: Byte |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | The data is used to display the number of bytes available in the dynamic memory. The data cannot be defined. The display is updated with every NCK power-up. Possible step-by-step procedure for calculating the memory requirement:<br>– Increase the input value by 1<br>– Power up the NCK<br>– Read off the memory requirement<br>– Calculate possible increase<br>The content of the machine data specifies the amount of dynamic RAM currently available via MD for the expansion of the volatile user data areas. Before expanding a parameter, such as the number of local user data (LUD), it is advisable to check that sufficient memory is available. | | | | |
| Special cases: | If more dynamic memory is requested than is currently available, the SRAM is deleted on the next power-up and all machine data are initialized with the default settings. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18060**<br>MD number | **INFO_FREE_MEM_STATIC**<br>Display data of free static memory | | | | |
| Default setting: 2621440<br>NCU573: 2621440<br>840Di: 524288<br>810D: 262144 | | Minimum input limit: – | | Maximum input limit: – | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: Byte |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | The contents of the machine data indicate how much non-volatile memory is available for the passive file system **at the time of the power-up**. Then the value is **no** longer **updated**.<br>To determine the current value at any given time the following operations are available:<br>MMC102: Services or programming: Free memory NCK<br>MMC100: Programming; softkey memory info.<br><br>If MDs that influence the amount of backed-up memory required are altered, then the amount of memory available for the passive file system also changes since the amount of memory allocated to the passive file system consists of the memory setting in MD 18230: MM_USER_MEM_BUFFERED (SRAM user memory) minus all other backup user data (see also description of MD 18350: MM_USER_FILE_MEM_MINIMUM (minimum part program memory)).<br>The data cannot be written. The display is only updated after every NCK power-up. | | | | |
| Special cases: | If more static memory is requested than is currently available, the SRAM is deleted on the next power-up and all machine data are initialized with the default settings. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18070**<br>MD number | **INFO_FREE_MEM_DPR**<br>Display data of free memory in DUAL_PORT RAM | | | | |
| Default setting: 0 | | Minimum input limit: – | | Maximum input limit: – | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: Byte |
| Data type: DWORD | | | | Applies from SW: | |
| Meaning: | None | | | | |
| MD irrelevant for ...... | The functionality is not available with SW 2. | | | | |

<table>
<tr><td>18080<br>MD number</td><td colspan="2">MM_TOOL_MANAGEMENT_MASK<br>Screen form for reserving memory for TM function</td></tr>
<tr><td>Default setting: 00H</td><td>Minimum input limit: 00H</td><td>Maximum input limit: FFFFH</td></tr>
<tr><td>Changes effective after POWER ON</td><td>Protection level: 1</td><td>Unit: –</td></tr>
<tr><td>Data type: DWORD</td><td colspan="2">Applies from SW: 2</td></tr>
<tr><td>Meaning:</td><td colspan="2">Step-by-step TM-specific memory reservation defined in particular by the following MD:<br>• MD 18086: MM_NUM_MAGAZINE_LOCATION<br>• MD 18084: MM_NUM_MAGAZINE<br>• MD 18096: MM_NUM_CC_TOA_PARAM<br>• MD 18094: MM_NUM_CC_TDA_PARAM<br>• MD 18098: MM_NUM_CC_MON_PARAM<br>• MD 18092: MM_NUM_CC_MAGLOC_PARAM<br>• MD 18090: MM_NUM_CC_MAGAZINE_PARAM<br><br>is implemented on a bit-coded basis as a function of this data.<br><br>Memory cannot be reserved simply by presetting the individual memory-configuring machine data. The memory configuration is not changed until the appropriate machine data is activated during the next power ON.<br><br>Bit 1: Make tool management data available:<br><br>• Memory-reserving MD for basic functionality of tools must be set:<br>MD 18092: MM_NUM_TOOL<br>MD 18100: MM_NUM_CUTTING_EDGES_IN_TOA<br><br>• Memory-reserving MD for tool management function must be set:<br>MD 18086: MM_NUM_MAGAZINE_LOCATION<br>MD 18084: MM_NUM_MAGAZINE<br><br>When bit 1 is set, TM-specific memory is added to the memory space defined in MM_NUM_TOOL.<br><br>Bit 2: Make tool monitoring data available:<br>• Memory-reserving MD for basic functionality of tools must be set:<br><br>MD 18092: MM_NUM_TOOL<br>MD 18100: MM_NUM_CUTTING_EDGES_IN_TOA<br><br>When bit 2 is set, memory for monitoring data is made available. TM-specific memory is added to the memory space programmed in MD 18100: MM_NUM_CUTTING_EDGES_IN_TOA.<br><br>Bit 3: OEM/CC data available:<br><br>• Memory-reserving MD must be set:<br>MD: MM_NUM_CC_...<br><br>When bit 3 is set, memory is made available for OEM applications.<br><br>Bit 4: Consider adjacent location tool management:<br><br>• Make memory available for TM function “Consider adjacent location”</td></tr>
<tr><td>Special cases, errors, ......</td><td colspan="2">The buffered data are lost if this machine data is altered!</td></tr>
<tr><td>Related to ....</td><td colspan="2">MD 18084: MM_NUM_MAGAZINE (number of magazines that the NCK can manage)<br>MD 18086: MM_NUM_MAGAZINE_LOCATION (number of magazine locations that the NCK can manage)<br>MD 18090: MM_NUM_CC_MAGAZINE_PARAM (number of magazine data that are set up and evaluated by the CC)<br>MD 18092: MM_NUM_CC_MAGLOC_PARAM (number of magazine location data that are set up and evaluated by the CC)<br>MD 18094: MM_NUM_CC_TDA_PARAM ((number of tool-specific data per tool for OEM and compile cycle)<br>MD 18096: MM_NUM_CC_TOA_PARAM (number of data per tool cutting edge for OEM and compile cycle)<br>MD 18098: MM_NUM_CC_MON_PARAM (number of monitoring data per tool cutting edge for OEM and compile cycle)</td></tr>
<tr><td>References:</td><td colspan="2">/FBW/, “Description of Functions Tool Management”</td></tr>
</table>

| 18082<br>MD number | MM_NUM_TOOL<br>Number of tools managed by the NCK | |
|---|---|---|
| Default setting: 30 | Minimum input limit: 0 | Maximum input limit: 600 |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 2 | |
| Meaning: | The NC cannot manage more tools than the maximum number entered in the MD. One tool has at least one cutting edge. | |
| Special cases, errors, ...... | The maximum number of possible tools corresponds to the number of cutting edges.<br>This MD must be set even if no tool management function is used.<br>The buffered data are lost if this machine data is altered! | |
| Related to .... | MD 18100: MM_NUM_CUTTING_EDGES_IN_TOA (number of tool offsets in NCK) | |

| 18084<br>MD number | MM_NUM_MAGAZINE<br>Number of magazines managed by NC | |
|---|---|---|
| Default setting: 3 | Minimum input limit: 0 | Maximum input limit: 32 |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 2 | |
| Meaning: | Number of magazines which NCK can manage.<br>The MD for TM MD 20310: TOOL_MANAGEMENT_MASK and MD 18080: MM_TOOL_MANAGEMENT_MASK and the optional TM $ON_TECHNO_FUNCTION_MASK must be set. | |
| MD irrelevant for ...... | MD is irrelevant if the tool management function is not in use. | |
| Special cases, errors, ...... | Only tool management stage 2:<br>Value = 0 –> tool management stage 2 cannot be activated because no memory area has been set up for the data.<br>The buffered data are lost if this machine data is altered! | |
| Related to .... | MD 18080: MM_TOOL_MANAGEMENT_MASK (mask for reserving memory for TM function)<br>MD 20310: TOOL_MANAGEMENT_MASK (activation of different variants of tool management function)<br>$ON_TECHNO_FUNCTION_MASK | |
| References | /FBW/, “Description of Functions Tool Management” | |

| 18086<br>MD number | MM_NUM_MAGAZINE_LOCATION<br>Number of magazine locations | |
|---|---|---|
| Default setting: 30 | Minimum input limit: 0 | Maximum input limit: 600 |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 2 | |
| Meaning: | Number of magazine locations which NCK can manage.<br>The MD for TM MD 20310: TOOL_MANAGEMENT_MASK and MD 18080: MM_TOOL_MANAGEMENT_MASK and the optional TM $ON_TECHNO_FUNCTION_MASK must be set. | |
| MD irrelevant for ...... | MD is irrelevant if the tool management function is not in use. | |
| Special cases, errors, ...... | Only tool management stage 2:<br>Value = 0 –> tool management stage 2 cannot be activated because no memory area has been set up for the data.<br>The buffered data are lost if this machine data is altered! | |
| Related to .... | MD 18080: MM_TOOL_MANAGEMENT_MASK (mask for reserving memory for TM function)<br>MD 20310: TOOL_MANAGEMENT_MASK (activation of different variants of tool management function)<br>$ON_TECHNO_FUNCTION_MASK | |
| References | /FBW/, “Description of Functions Tool Management” | |

| | | | | | |
|---|---|---|---|---|---|
| **18090**<br>MD number | **MM_NUM_CC_MAGAZINE_PARAM**<br>Compile cycles of tool management: Number of magazine data | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 10 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 2 | |
| Meaning: | Only if MD for tool management and tool management option are set:<br><br>Number of magazine data (format IN_int) for which a memory area is set up and which can be evaluated by compile cycles. | | | | |
| MD irrelevant for ...... | MD is irrelevant if tool management function is not activated. | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | |
| Related to .... | MD 18080: MM_TOOL_MANAGEMENT_MASK (mask for reserving memory for TM function)<br>MD 18084: MM_NUM_MAGAZINE (number of magazines managed by the NC) | | | | |
| References: | /FBW/, “Description of Functions Tool Management” | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18092**<br>MD number | **MM_NUM_CC_MAGLOC_PARAM**<br>Compile cycles of tool management: Number of magazine location data | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 10 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 2 | |
| Meaning: | Only if MD for tool management and tool management option are set:<br><br>Number of magazine data (format IN_int) for which a memory area is set up and which can be evaluated by compile cycles. | | | | |
| MD irrelevant for ...... | MD is irrelevant if tool management function is not activated. | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | |
| Related to .... | MD 18080: MM_TOOL_MANAGEMENT_MASK (mask for reserving memory for TM function) | | | | |
| References: | /FBW/, “Description of Functions Tool Management” | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18094**<br>MD number | **MM_NUM_CC_TDA_PARAM**<br>Compile cycles of tool management: Number of TDA data | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 10 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 2 | |
| Meaning: | Only if MD for tool management and tool management option are set:<br><br>Number of TDA (tool-specific) data (format IN_int) for which a memory area is set up and which can be evaluated by compile cycles. | | | | |
| MD irrelevant for ...... | MD is irrelevant if tool management function is not activated. | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | |
| Related to .... | MD 18080: MM_TOOL_MANAGEMENT_MASK (mask for reserving memory for TM function)<br>MD 18082: MM_NUM_TOOL (number of tools managed by the NCK) | | | | |
| References: | /FBW/, “Description of Functions Tool Management” | | | | |

| 18096<br>MD number | **MM_NUM_CC_TOA_PARAM**<br>Compile cycles of tool management: Number of TOA data | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 10 |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 2 | |
| Meaning: | Only if MD for tool management and tool management option are set:<br>Number of TOA (tool-specific) data (format IN_int) per cutting edge for which a memory area is set up and which can be evaluated by compile cycles. | |
| MD irrelevant for ...... | Tool management stages 1 and 2 not activated. | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | |
| Related to .... | MD 18080: MM_TOOL_MANAGEMENT_MASK (mask for reserving memory for TM function)<br>MD 18100: MM_NUM_CUTTING_EDGES_IN_TOA (number of tool offsets in NCK) | |
| References: | /FBW/, “Description of Functions Tool Management” | |

| 18098<br>MD number | **MM_NUM_CC_MON_PARAM**<br>Compile cycles of tool management: Number of monitor data | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 10 |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 2 | |
| Meaning: | For tool management compile cycles:<br>Number of monitor data which are created for each tool and which can be evaluated by compile cycles. | |
| MD irrelevant for ...... | MD is irrelevant if tool management function is not activated. | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | |
| Related to .... | MD 18080: MM_TOOL_MANAGEMENT_MASK (mask for reserving memory for TM function)<br>MD 18100: MM_NUM_CUTTING_EDGES_IN_TOA (number of tool offsets in NCK) | |
| References: | /FBW/, “Description of Functions Tool Management” | |

| 18100<br>MD number | **MM_NUM_CUTTING_EDGES_IN_TOA**<br>Number of tool offsets in NCK | |
|---|---|---|
| Default setting: 30 | Minimum input limit: 0 | Maximum input limit: 1500 |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 1.1 | |
| Meaning: | The MD specifies the number of tool edges in the NCK. This machine data reserves approximately 250 bytes of backup memory per TOA module for each tool edge, irrespective of the tool type.<br>Tools with cutting edge types 400–499 (= grinding tools) also occupy the location of a cutting edge.<br>E.g.:<br>Define 10 grinding tools with one cutting edge each.<br>At least the following must apply:<br>MM_NUM_TOOL = 10<br>MM_NUM_CUTTING_EDGES_IN_TOA = 20<br>See also MM_NUM_TOOL | |
| Special cases, errors, ...... | The data in the buffer are lost when the machine data are changed! | |
| References: | /FBW/, “Description of Functions Tool Management” | |

| | | | |
|---|---|---|---|
| **18118**<br>MD number | **MM_NUM_GUD_MODULES**<br>Number of GUD modules | | |
| Default setting: 3 | Minimum input limit: 1 | Maximum input limit: 9 | |
| Changes effective after POWER ON | Protection level: 2 | Unit: – | |
| Data type: DWORD | Applies from SW: 2 | | |
| Meaning: | A GUD block corresponds to a file in which user-defined data can be stored. 9 GUD blocks are available of which 3 are already assigned to specific users/applications.<br>UGUD_DEF_USER (block for user)<br>SGUD_DEF_USER (block for SIEMENS)<br>MGUD_DEF_USER (block for machine manufacturer) | | |
| Special cases, errors, ...... | The number of GUD modules is determined by the GUD with the highest number.<br>Example: If the following GUD modules are defined,<br>UGUD<br>MGUD<br>GUD5<br>GUD8<br>then the machine data must be set to a value of 8, signifying<br>a memory requirement of 8 x 120 bytes = 960 bytes.<br><br>It is therefore advisable to select the “lowest” possible GUD module. If GUD modules UGUD and MGUD have not been assigned elsewhere, then they may be used for this purpose. | | |
| Related to: | MD 18150: MM_GUD_VALUES_MEM (memory for user variables) | | |

| | | | |
|---|---|---|---|
| **18120**<br>MD number | **MM_NUM_GUD_NAMES_NCK**<br>Number of global user variables | | |
| Default setting: 10 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after POWER ON | Protection level: 2 | Unit: – | |
| Data type: DWORD | Applies from SW: 1.1 | | |
| Meaning: | Defines the number of user variables for NCK global user data (GUD). Approximately 80 bytes of memory per variable are reserved in the SRAM for the name of the variable. The additional memory required for the value of the variable depends on the data type of the variable. The number of available NCK-global user variables is restricted by the limit value set in MM_NUM_GUD_NAMES_NCK or MD 18150: MM_GUD_VALUES_MEM (memory for user variables). | | |
| Special cases, errors, ...... | The data in the buffer are lost when the machine data are changed! | | |
| Related to .... | MD 18150: MM_GUD_VALUES_MEM (memory for user variables) | | |

| | | | |
|---|---|---|---|
| **18130**<br>MD number | **MM_NUM_GUD_NAMES_CHAN**<br>Number of channel-specific user variables | | |
| Default setting: 10 | Minimum input limit: 0 | Maximum input limit: plus | |
| Changes effective after POWER ON | Protection level: 2 | Unit: – | |
| Data type: DWORD | Applies from SW: 1.1 | | |
| Meaning: | Defines the number of user variables for channel-specific global user data (GUD). Approximately 80 bytes of memory per variable are reserved in the SRAM for the name of the variable. The additional memory required for the variable value is equal to the size of the data type of the variable multiplied by the number of channels. This means that each channel has its own memory available for the variable values. The number of channel-specific, global user variables available is exhausted when the limit defined in MD 18130: MM_NUM_GUD_NAMES_CHAN or MD 18150: MM_GUD_VALUES_MEM (memory for user variables). | | |
| Special cases, errors, ...... | The data in the buffer are lost when the machine data are changed! | | |
| Related to .... | MD 18150: MM_GUD_VALUES_MEM (memory for user variables) | | |

| | | | | | |
|---|---|---|---|---|---|
| **18140**<br>MD number | **MM_NUM_GUD_NAMES_AXIS**<br>Number of axis-specific user variables | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: | |
| Special cases, errors, ...... | The data in the buffer are lost when the machine data are changed! | | | | |
| MD irrelevant for ... ... | The functionality is not available with SW 2. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18150**<br>MD number | **MM_GUD_VALUES_MEM**<br>Memory location for user variables | | | | |
| Default setting: 2 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: KB |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | The specified value reserves memory for the variable values of the global user data (GUD). The dimensioning of the memory depends to a large extent on the data types used for the variables.<br>Overview of memory used by data types:<br><br>Data type Memory used<br>REAL 8 bytes<br>INT 4 bytes<br>BOOL 1 byte<br>CHAR 1 byte<br>STRING 1 1 byte per character, 100 characters permitted per string<br>AXIS 4 bytes<br>FRAME 400 bytes<br><br>The total memory required by channel or axis-specific global user variables is the memory used by the variables multiplied by the number of channels or axes.<br>The number of global user variables available is exhausted when the limits defined in the MD: MM_NUM_GUD_NAMES_xxxx or MM_GUD_VALUES_MEM are reached.<br>The battery-backed memory is used. | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | |
| Related to .... | MD 18118: MM_NUM_GUD_MODULES (number of GUD modules)<br>MD 18120: MM_NUM_GUD_NAMES_NCK (number of global user variables)<br>MD 18130: MM_NUM_GUD_NAMES_CHAN (number of channel-specific user variables) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18160**<br>MD number | **MM_NUM_USER_MACROS**<br>Number of macros | | | | |
| Default setting: 10 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | Defines the total number of macros which can be stored in the files _N_SMAC_DEF, _N_NMAC_DEF and _N_UMAC_DEF. When opened, each of these files requires at least one KB of part program memory for the file code. If this limit for the file code is exceeded, another KB is reserved for the file.<br>Dynamic user memory is used. Approximately 375 bytes per macro are reserved for the specified number of macros for management tasks. | | | | |
| Special cases, errors, ...... | The data in the buffer are lost when the machine data are changed! | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18170**<br>MD number | **MM_NUM_MAX_FUNC_NAMES**<br>Number of miscellaneous functions | | | | |
| Default setting: 30 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | The machine data limits the maximum number of miscellaneous functions over and above the predefined functions (such as sine, cosine, etc.) which can be used in<br>– cycle programs<br>– compile cycle software.<br>The function names are entered in the global NCK dictionary and may not conflict with the names that already exist.<br>The SIEMENS cycle package of SW version 1 contains miscellaneous functions that are taken into account by the default setting of the MD.<br>The data are stored in volatile memory. Approximately 150 bytes are required for each miscellaneous function for management purposes. | | | | |
| Related to .... | MD 18180: MM_NUM_MAX_FUNC_PARAM (no. of miscellaneous function parameters) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18180**<br>MD number | **MM_NUM_MAX_FUNC_PARAM**<br>Number of additional parameters | | | | |
| Default setting: 300 | | Minimum input limit: 0 | | Maximum input limit: plus | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | Defines the maximum number of parameters required for the miscellaneous functions in<br>– cycle programs<br>– compile cycle software.<br>50 parameters are required for the miscellaneous functions of the Siemens cycle package of SW version 1.<br>The data are stored in volatile memory. Approximately 72 bytes are reserved for each parameter. | | | | |
| Related to .... | MD 18170: MM_NUM_MAX_FUNC_NAMES (number of miscellaneous functions) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18190**<br>MD number | **MM_NUM_PROTECT_AREA_NCK**<br>Number of protection zones in NCK | | | | |
| Default setting:<br>0 | | Minimum input limit:<br>0 | | Maximum input limit:<br>10 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 2 | |
| Meaning: | This machine data defines how many blocks are created for the protection zones available in the NCK. | | | | |
| Special cases, errors, ...... | The data in the buffer are lost when the machine data are changed! | | | | |
| References | /FB/, A3, “Axis Monitoring, Protection Zones” | | | | |

| **18210**<br>MD number | **MM_USER_MEM_DYNAMIC**<br>User memory in DRAM | |
|---|---|---|
| Default setting:<br>1000 | Minimum input limit:<br>– | Maximum input limit:<br>– |
| Changes effective after POWER ON | Protection level: 2/7 | Unit: KB |
| Data type: DWORD | Applies from SW: 1.1 | |
| Meaning: | The DRAM which physically exists in the NC is shared by the system and the user.<br>MM_USER_MEM_DYNAMIC defines the amount of memory in the DRAM that is available to the user. The input limits are dependent on the hardware and software configuration of the CNC.<br>This memory area contains various types of user data such as<br>– local user data<br>– REORG-LOG data.<br>The data in the dynamic memory are not backed up.<br>The input limits ensure that the memory space reserved does not exceed the amount of memory which is actually available in the hardware. | |
| Application example(s) | When the default values are set, the following DRAM memory is available to the user with the NCU 572/573 depending on the number of defined channels:<br>– 1 MB (1 channel defined)<br>– 300 KB (2 channels defined). | |
| Special cases, errors, ... ... | During power-up, the system software compares the total demands for DRAM with the value set in MD: MM_USER_MEM_DYNAMIC.<br>If the memory required exceeds the capacity defined in the machine data, alarm 6000 “Memory allocation with standard machine data” is output.<br>Alarm 6030 “User memory limit has been adjusted” is output if the control system detects during power-up that the memory capacity requested through MM_USER_MEM_DYNAMIC is greater than the physical memory size. | |
| Related to: | The available dynamic memory is displayed in MD 18050: INFO_FREE_MEM_DYNAMIC (display data for available DRAM). | |

| **18220**<br>MD number | **MM_USER_MEM_DPR**<br>User memory in dual-port RAM | |
|---|---|---|
| Default setting: 0 | Minimum input limit: – | Maximum input limit: – |
| Changes effective after POWER ON | Protection level: 0 | Unit: KB |
| Data type: DWORD | Applies from SW: | |
| MD irrelevant for ...... | The functionality is not available with SW 2. | |

<table>
<tr><td colspan="2">18230<br>MD number</td><td colspan="4">MM_USER_MEM_BUFFERED<br>User memory in SRAM</td></tr>
<tr><td colspan="2">Default setting:<br>280</td><td colspan="2">Minimum input limit:<br>–</td><td colspan="2">Maximum input limit:<br>–</td></tr>
<tr><td colspan="3">Changes effective after POWER ON</td><td colspan="2">Protection level: 1/7</td><td>Unit: KB</td></tr>
<tr><td colspan="3">Data type: DWORD</td><td colspan="3">Applies from SW: 1.1</td></tr>
<tr><td>Meaning:</td><td colspan="5">Defines the size of the battery-backed user memory. Various types of user data are stored in this area such as, for example:<br>– NC part programs<br>– R parameters<br>– Tool data<br>– User macros<br>– Global user data<br><br>The settable values depend on the hardware and software configuration.<br>512 KB are available in the hardware for the NCU 571.<br>512 KB or 2 MB are available for the NCU 572/573 depending on the hardware configuration.<br>The CNC requires approximately 30 KB of this for its operating system, leaving 480 KB. Some of this remaining memory is allocated for further areas permanently reserved for machine data, setting data and data management.<br>The CNC manufacturer guarantees 256 KB user memory in the SRAM.<br>The availability of more than 256 KB user memory cannot be guaranteed in conjunction with the following software versions.<br><br>SRAM with 2 MB:<br><br>If the NCU 572/573 is used with a larger memory, then the memory must be released.<br><br>• Enter the value 1900 in MD 18230<br>• Make a copy of a series start-up file<br>• Execute power ON (in order to reorganize the memory)<br>• Load the series start-up file back into the control system<br><br>During power-up, the system software compares the total amount of battery-backed memory required with the value set in MD 18230: MM_USER_MEM_BUFFERED.<br>If the memory required exceeds the capacity defined in the machine data, alarm 6000 “Memory allocation with standard machine data” is output. Alarm 6030 “User memory limit has been adjusted” is output if the control system detects during power-up that the memory capacity requested in MD 18230: MM_USER_MEM_BUFFERED is larger than the physical memory.</td></tr>
<tr><td>Special cases, errors, ......</td><td colspan="5">The data in the buffer are lost when the machine data are changed!</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| **18240**<br>MD number | **MM_LUD_HASH_TABLE_SIZE**<br>Hash table size for user variables | | | | |
| Default setting: 11 | | Minimum input limit: 3 | | Maximum input limit: 107 | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: prime number |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | Defines the memory size for local user data (LUD). The value entered must be a prime number. The setting allows the optimization of<br>– the interpreter execution time (lower value = longer execution time) and<br>– memory requirements (lower value = less memory).<br>A larger table requires a smaller number of decoding operations for internal decoding of the variables and consequently a shorter interpreter execution time. The value set in this machine data affects the amount of dynamic memory required for the management of the modules for the local user variables with REORG, see MD 28010: MM_NUM_REORG_LUD_MODULES (number of modules for local user variables with REORG (DRAM)). | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | |

| 18242<br>MD number | MM_MAX_SIZE_OF_LUD_VALUE<br>Maximum field size of LUD variables | | | | |
|---|---|---|---|---|---|
| Default setting: 8192<br>Default setting: 496 (SW4.1 and higher)<br>LUD / GUD up to 12 axes: 660<br>LUD / GUD > 12 axes: 920<br>NC memory GUD / LUD: – | | Min. input limit: 128<br>240 (from SW4.1)<br>–<br>–<br>240 | | Max. input limit: 8192<br>496 (from SW4.1)<br>–<br>–<br>8192 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: Byte |
| Data type: DWORD | | | | Applies from SW: 2 | |
| Meaning: | MD 18242: MM_MAX_SIZE_OF_LUD_VALUE specifies the block size in which the total memory defined in<br>MD 28040: MM_LUD_VALUES_MEM is assigned to the part programs of the channel.<br><br>The first variable to occur in the part program occupies a block of the size specified in MD 18242: MM_MAX_SIZE_OF_LUD_VALUE. The following variables are also stored in this block. If the block is full of values or cannot accommodate any further variable, then another block is requested.<br><br>MD 18242: MM_MAX_SIZE_OF_LUD_VALUE must be set to the same value as the memory required by the largest possible variable used.<br><br>Data type — Memory used<br>REAL — 8 bytes<br>INT — 4 bytes<br>BOOL — 1 byte<br>CHAR — 1 byte<br>STRING — 1 byte per character, 100 characters permitted per string<br>AXIS — 4 bytes<br>FRAME — 400 bytes | | | | |
| Related to .... | MD 28040: MM_LUD_VALUES_MEM (memory size for local user variables (DRAM)) | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | |

| 18250<br>MD number | MM_CHAN_HASH_TABLE_SIZE<br>Hash table size for channel-specific data | | | | |
|---|---|---|---|---|---|
| Default setting: 7 | | Minimum input limit: 7 | | Maximum input limit: 193 | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: prime number |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | Defines the size for channel-specific names. The value entered must be a prime number.<br>The setting allows the optimization of<br>– the interpreter execution time (lower value = longer execution time) and<br>– memory requirements (lower value = less dynamic memory).<br>A larger table requires a smaller number of decoding operations for internal decoding of the variables and consequently a shorter interpreter execution time. The value of this machine data affects the amount of dynamic memory required. The required memory for each channel in bytes is equal to the value entered multiplied by 68. | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | |

| 18260<br>MD number | MM_NCK_HASH_TABLE_SIZE<br>Hash table size for global data | | |
|---|---|---|---|
| Default setting: 2503 | Minimum input limit: 537 | Maximum input limit: 4001 | |
| Changes effective after POWER ON | Protection level: 0 | Unit: prime number | |
| Data type: DWORD | Applies from SW: 1.1 | | |
| Meaning: | Defines the size for NCK-specific names. The value entered must be a prime number. The setting allows the optimization of<br>– the interpreter execution time (lower value = longer execution time) and<br>– memory requirements (lower value = less memory).<br>A larger table requires a smaller number of decoding operations for internal decoding of the variables and consequently a shorter interpreter execution time. The value of this machine data affects the amount of dynamic memory required. The required memory in bytes is equal to the value entered multiplied by 68. | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | |

| 18270<br>MD number | MM_NUM_SUBDIR_PER_DIR<br>Number of subdirectories | | |
|---|---|---|---|
| Default setting:<br>30 | Minimum input limit:<br>24 | Maximum input limit:<br>250 | |
| Changes effective after POWER ON | Protection level: 1 | Unit: – | |
| Data type: DWORD | Applies from SW: 1.1 | | |
| Meaning: | Defines the maximum number of subdirectories that a directory in the passive file system can have. The number of directories is limited by MD 18310: MM_NUM_DIR_IN_FILESYSTEM (no. of directories in passive file system). The memory requirement is contained in the memory for the number of files per directory (see MD 18260: MM_NUM_FILES_PER_DIR). | | |
| Related to .... | MD 18310: MM_NUM_DIR_IN_FILESYSTEM (no. of directories in passive file system) | | |

| 18280<br>MD number | MM_NUM_FILES_PER_DIR<br>Number of files per directory | | |
|---|---|---|---|
| Default setting:<br>100 | Minimum input limit:<br>64 | Maximum input limit:<br>512 | |
| Changes effective after POWER ON | Protection level: 1 | Unit: – | |
| Data type: DWORD | Applies from SW: 1.1 | | |
| Meaning: | Specifies the maximum number of files which can be created in a directory or subdirectory of the passive file system. The total number of files is limited by MD 18320: MM_NUM_FILES_IN_FILESYSTEM (no. of files in passive file system). The memory in bytes required for the management of files in the directory is the value entered multiplied by 40. The highest value of MD 18280: MM_NUM_FILES_PER_DIR (number of files per directory) and MD 18270: MM_NUM_SUBDIR_PER_DIR (no. of subdirectories) must be entered as the MD setting. The memory required to manage files in the passive file system is reserved by MD 18320: MM_NUM_FILES_IN_FILESYSTEM. | | |
| Special cases, errors, ..... | The buffered data are lost if this machine data is altered!<br>Note:<br>An alteration of the MD has an effect on directories created after this. This means that if the number of files in an existing directory is to be altered, the existing directory must first be deleted and then a new directory must be created (but only after having first saved the files)! | | |
| Related to .... | MD 18320: MM_NUM_FILES_IN_FILESYSTEM (number of files in passive file system) | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **18290** MD number | **MM_FILE_HASH_TABLE_SIZE** Hash table size for files in a directory | | | | | |
| Default setting: 19 | | Minimum input limit: 3 | | | Maximum input limit: 299 | |
| Changes effective after POWER ON | | | Protection level: 0 | | | Unit: prime number |
| Data type: DWORD | | | | Applies from SW: 1.1 | | |
| Meaning: | Defines the size of the hash table for the files of a directory. The value entered must be a prime number. The setting allows the optimization of<br>– the interpreter execution time (lower value – longer execution time) and<br>– memory requirements (lower value = less memory).<br>The value of this machine data affects the amount of static memory required for the management of directories, see MD 18310: MM_NUM_DIR_IN_FILESYSTEM (no. of directories in passive file system) | | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **18300** MD number | **MM_DIR_HASH_TABLE_SIZE** Hash table size for subdirectories | | | | | |
| Default setting: 7 | | Minimum input limit: 3 | | | Maximum input limit: 349 | |
| Changes effective after POWER ON | | | Protection level: 0 | | | Unit: prime number |
| Data type: DWORD | | | | Applies from SW: 1.1 | | |
| Meaning: | Defines the size of the hash table for the subdirectories of a directory. The value entered must be a prime number. The setting allows the optimization of<br>– the interpreter execution time (lower value – longer execution time) and<br>– memory requirements (lower value = less memory).<br>The value of this machine data affects the amount of static memory required for the management of directories, see MD 18310: MM_NUM_DIR_IN_FILESYSTEM (no. of directories in passive file system) | | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **18310** MD number | **MM_NUM_DIR_IN_FILESYSTEM** Number of directories in passive file system | | | | | |
| Default setting: 30 | | Minimum input limit: 30 | | | Maximum input limit: 256 | |
| Changes effective after POWER ON | | | Protection level: 2 | | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 1.1 | | |
| Meaning: | This machine data limits the number of directories in the passive file system and can be used to reserve memory in the SRAM for the management of the directories. The directories and subdirectories of the passive file system set up by the system are included in this machine data. The memory required for the management of the directories can be calculated as follows:<br>Memory required = a (440+28 (b+c)) bytes<br>a = Input value of MD 18310: MM_NUM_DIR_IN_FILESYSTEM (no. of directories in passive file system)<br>b = Input value of MD 19300: MM_DIR_HASH_TABLE_SIZE (HASH table size for subdirectories)<br>c = Input value of MD 18290: MM_FILE_HASH_TABLE_SIZE (hash table size for the files of a directory) | | | | | |
| Special cases, errors, ...... | The data in the buffer are lost when the machine data are changed! | | | | | |
| Related to .... | MD 18270: MM_NUM_SUBDIR_PER_DIR (no. of subdirectories) | | | | | |

| 18320<br>MD number | MM_NUM_FILES_IN_FILESYSTEM<br>Number of files in passive file system | |
|---|---|---|
| Default setting:<br>100 | Minimum input limit:<br>64 | Maximum input limit:<br>512 |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 1.1 | |
| Meaning: | Defines the number of files available in the part program memory. This machine data is used to reserve memory in SRAM – approximately 320 bytes – for file management. Each file created requires a minimum of one KB of memory for the file code. If the one KB limit for the file code is exceeded another KB is reserved for the file. | |
| Special cases, errors, ...... | The data in the buffer are lost when the machine data are changed! | |
| Related to: | MD 18280: MM_NUM_FILES_PER_DIR (number of files in directories) | |

| 18342<br>MD number | MM_CEC_MAX_POINTS<br>Maximum table size for sag compensation | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 2000 |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 2 | |
| Meaning: | Maximum table size for interpolative compensation between axes.<br><br>When MM_CEC_MAX_POINTS = 0, no memory is set up for the table. The sag compensation function cannot then be used. | |
| Special cases: | A change in this machine data causes reconfiguration of the buffered memory area. | |

| 18350<br>MD number | MM_USER_FILE_MEM_MINIMUM<br>Minimum NC program memory | |
|---|---|---|
| Default setting: 20 | Minimum input limit: 20 | Maximum input limit: 100 |
| Changes effective after POWER ON | Protection level: 1 | Unit: KB |
| Data type: DWORD | Applies from SW: 1.1 | |
| Meaning: | Defines the minimum backup memory area remaining for the files of the passive file system. The settable value depends on the hardware and software configurations (SRAM memory allocation) and on MD 18230: MM_USER_MEM_BUFFERED (SRAM user memory). When the SRAM memory is allocated, the remaining memory is allocated to the files of the passive file system. In order to ensure that the file system can operate, the memory space specified in MM_USER_FILE_MEM_MINIMUM must be available to the memory. If this condition is not met, the memory is allocated with the default data on the control and all the data stored in the SRAM by the user is lost. Alarm 6000 “Memory allocation with standard machine data” is also output.<br>The available part program memory capacity is displayed in MD 18060: INFO_FREE_MEM_STATIC (display of free static memory). | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered and the remaining memory is smaller than the value in MM_USER_FILE_MEM_MINIMUM. | |

| | | | | | |
|---|---|---|---|---|---|
| **18500**<br>MD number | **MM_EXTCOM_TASK_STACK_SIZE**<br>Stack size for external communication task | | | | |
| Default setting: 17 | | Minimum input limit: 4 | | Maximum input limit: 20 | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: KB |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | The size of the stack for external communication. The dynamic memory area is used. | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18510**<br>MD number | **MM_SERVO_TASK_STACK_SIZE**<br>Stack size of servo task | | | | |
| Default setting: 8 | | Minimum input limit: 4 | | Maximum input limit: 20 | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: KB |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | Defines the stack size for the SERVO task. The dynamic memory is used for this purpose. | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **18520**<br>MD number | **MM_DRIVE_TASK_STACK_SIZE**<br>Stack size of drive task | | | | |
| Default setting: 8 | | Minimum input limit: 4 | | Maximum input limit: 20 | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: KB |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | The stack size for the SIMODRIVE task is stored in this data.<br>The stack is set up in dynamic memory. | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | |

## 4.2 Channelspecific machine data

| | | | | |
|---|---|---|---|---|
| **27900**<br>MD number | **REORG_LOG_LIMIT**<br>Percentage of IPO buffer for log file enable | | | |
| Default setting: 1 | Minimum input limit: – | | Maximum input limit: – | |
| Changes effective after POWER ON | | Protection level: 0 | | Unit: – |
| Data type: BYTE | | | Applies from SW: 1.1 | |
| Meaning: | The machine data defines the percentage of the IPO buffer above which data in the REORG LOG memory can be released in stages, if the block preparation has been interrupted due to an overflow of the REORG LOG data memory. The released data are no longer available to the REORG function (References: /FB/, K1, “Mode Groups, Channels, Program Operation Mode”). A consequence of this status is that a further REORG command is canceled with an error message. If the status of non-reorganizability occurs, warning 15110 is output. The output of the warning can be suppressed by enabling the highest significant bit. The bit is set by adding the value 128 to the input value in REORG_LOG_LIMIT.<br>In addition to the instructions of the NC blocks, the size of the IPO buffer and the REORG data memory also affect the frequency of data release. | | | |
| Related to .... | MD 28000: MM_REORG_LOG_FILE_MEM (memory size for REORG)<br>MD 28060: MM_IPO_BUFFER_SIZE (no. of blocks in the IPO buffer) | | | |

| | | | | |
|---|---|---|---|---|
| **28000**<br>MD number | **MM_REORG_LOG_FILE_MEM**<br>Memory size for REORG | | | |
| Default setting:<br>10 | Minimum input limit:<br>1 | | Maximum input limit:<br>500 | |
| Changes effective after POWER ON | | Protection level: 2 | | Unit: KB |
| Data type: DWORD | | | Applies from SW: 1.1 | |
| Meaning: | Defines the size of dynamic memory for the REORG LOG data. The size of the memory determines the amount of data available for the REORG function. | | | |
| References: | /FB/, K1, “Mode Group, Channels, Program Operation” | | | |

| | | | | |
|---|---|---|---|---|
| **28010**<br>MD number | **MM_NUM_REORG_LUD_MODULES**<br>Number of modules for local user variables with REORG | | | |
| Default setting: 4 | | Minimum input limit: 0 | | Maximum input limit: 100 |
| Changes effective after POWER ON | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 1.1 | |
| Meaning: | Defines the number of additional LUD modules provided for the REORG function (see Description of Functions, Mode Groups, Channels, Program Operation Mode (K1)). If the REORG function is not used, this value can be 0. The CNC always opens 12 LUD modules: 8 for NC programs and 4 for asynchronous subprograms. One LUD module is required for each NC program or asynchronous subprogram containing a definition of a local variable. It may be necessary to increase this value for REORG if a larger IPO buffer is provided and a large number of short NC programs containing LUD variable definitions are active (the NC blocks of the program are stored in prepared format in the IPO buffer. One LUD module is required for each of these programs. The capacity of the reserved memory is affected by the number of LUD per NC program and their individual memory requirements.<br>The LUD modules are stored in dynamic memory.<br>The memory required for managing the modules for local user variables with REORG can be calculated as follows:<br>Memory = a x (200 + b x 160) bytes<br>a = Total number of LUD modules = 8 + 4 + value in MD: MM_NUM_REORG_LUD_MODULES<br>b = Input value of MD 18240: MM_LUD_HASH_TABLE_SIZE (hash table size for user variables)<br>The size of the LUD modules depends on the number of active LUD and their data types.<br>The memory for LUD modules is limited by MD 28000: MM_REORG_LOG_FILE_MEM (memory size for REORG). | | | |
| Application example(s) | Example:<br>A main program consisting of 4 NC blocks is started:<br>– A LUD variable is defined in the first block.<br>– A subprogram, nested up to 8 levels, is called in each of the second and third blocks.<br>– The fourth block terminates the program.<br>Each subprogram comprises 3 NC blocks:<br>– An LUD variable is defined in the first block.<br>– A subprogram call to the next program level is executed in the second block.<br>– The third block terminates the subprogram.<br>Instead of a subprogram call, the subprogram in the last program level contains a different command, such as a traversing movement. This makes a total of 15 programs with 46 NC blocks which can all be stored in prepared format in the IPO buffer. Since the REORG function requires all the data of the 46 blocks, LUD modules for 3 programs are missing. A value of 3 for the additionally required LUD data blocks must be entered in MM_NUM_REORG_LUD_MODULES for the example given. | | | |

| 28020<br>MD number | MM_NUM_LUD_NAMES_TOTAL<br>Number of local user variables | |
|---|---|---|
| Default setting:<br>200 | Minimum input limit:<br>0.0 | Maximum input limit:<br>plus |
| Changes effective after POWER ON | Protection level: 2 | Unit: – |
| Data type: DWORD | Applies from SW: 1.1 | |
| Meaning: | Defines the number of variables for the local user data (LUD) which are permitted to exist in the active sections of the program. Approximately 150 bytes of memory per variable are reserved for the names of the variables and the variable value. The memory required for the variable value is equal to the size of the data type. If the total of the local user variables from the active main program and the related subprograms are larger than the defined limit, the variables which are over the limit are not accepted during execution of the program.<br>Dynamic memory is used to store the variable names and values.<br><br>Overview of the memory used by data types:<br>Data type Memory used<br>REAL 8 bytes<br>INT 4 bytes<br>BOOL 1 byte<br>CHAR 1 byte<br>STRING 1 byte per character, 200 characters per string<br>AXIS 4 bytes<br>FRAME 400 bytes | |

| 28040<br>MD number | MM_LUD_VALUES_MEM<br>Memory size for local user variables | |
|---|---|---|
| Default setting: 12, 12, ..., | Minimum input limit: 0.0 | Maximum input limit: plus |
| Changes effective after POWER ON | Protection level: 2 | Unit: KB |
| Data type: DWORD | Applies from SW: 4.3 | |
| Meaning: | This MD defines the amount of memory space available for LUD variables.<br>The number of available LUD is exhausted when one of the limit values in either MD 28020: MM_NUM_LUD_NAMES_TOTAL or MM_LUD_VALUES_MEM is reached.<br>The memory defined here is subdivided into (MM_LUD_VALUES_MEM * 1024) / MM_MAX_SIZE_OF_LUD_VALUE blocks and allocated to part programs which request memory. Each part program which contains at least one definition of LUD variables or which has call parameters uses at least one such block.<br>It should be remembered that several part programs can be open at once and thus use memory on the NCK. The number depends on the type of programming, the program length and the size of the internal NCK block memory upwards of (MM_IPO_BUFFER_SIZE, MM_NUM_BLOCKS_IN_PREP). | |
| Related to: | MD 28020: MM_NUM_LUD_NAMES_TOTAL (number of local user variables (DRAM)) | |

| 28050<br>MD number | MM_NUM_R_PARAM<br>Number of channel-specific R parameters | |
|---|---|---|
| Default setting:<br>100 | Minimum input limit: 0 | Maximum input limit: 32535 |
| Changes effective after POWER ON | Protection level: 0 / 0 | Unit: – |
| Data type: DWORD | Applies from SW: 4.3 | |
| Meaning: | Defines the number of R parameters available on the channel. A maximum of 32535 R parameters are available for each channel. This machine data is used to reserve 8 bytes of backup user memory for each R parameter. R parameters require substantially less management overhead compared with LUD and GUD variables. | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | |

| | | | | |
|---|---|---|---|---|
| **28060**<br>MD number | **MM_IPO_BUFFER_SIZE**<br>Number of NC blocks in the IPO buffer | | | |
| Default setting:<br>10, 10, 10, 10, 10, 10, 10, 10 | Minimum input limit:<br>2 | | Maximum input limit:<br><br>NCU 571: 300<br>NCU 572, 573: 300<br>810D: 180<br>810D_2: 300 | |
| Changes effective after POWER ON | | Protection level: 0 / 0 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 4.3 | |
| Meaning: | Defines the number of blocks in the interpolation buffer. This buffer contains prepared NC blocks which are provided for interpolation. Approximately 10 Kbytes of dynamic user memory is reserved for each NC block. The data also limits the number of Look Ahead blocks for limiting the speed in the Look Ahead function.<br><br>The MM_IPO_BUFFER_SIZE is set by the system. | | | |
| Related to: | MD 28070: MM_NUM_BLOCKS_IN_PREP (number of blocks for block preparation)<br>SD MAX_BLOCKS_IN_IPOBUFFER | | | |

| | | | | |
|---|---|---|---|---|
| **28070**<br>MD number | **MM_NUM_BLOCKS_IN_PREP**<br>Number of blocks for block preparation | | | |
| Default setting:<br>NCU 570: 38<br>NCU 571: 30<br>NCU 572: 38<br>NCU 573: 38<br>810D: 30<br>810D_2: 38 | Minimum input limit:<br>20 | | Maximum input limit:<br>NCU 570: ***<br>NCU 571: ***<br>NCU 572: ***<br>NCU 573: ***<br>810D: ***<br>810D_2: *** | |
| Changes effective after POWER ON | | Protection level: 0 / 0 | | Unit: Number of internal blocks |
| Data type: DWORD | | | Applies from SW: 4.3 | |
| Meaning: | Defines the number of blocks available for NC block preparation. This figure is determined mainly by the system software and is used for optimization. A part of dynamic memory is reserved (approximately 10 KB per NC block). | | | |
| Related to: | MD 28060: MM_IPO_BUFFER_SIZE (number of NC blocks with IPO buffer) | | | |

| | | | | |
|---|---|---|---|---|
| **28080**<br>MD number | **MM_NUM_USER_FRAMES**<br>Number of settable frames | | | |
| Default setting: 5 | Minimum input limit: 5 | | Maximum input limit: 100 | |
| Changes effective after POWER ON | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 1.1 | |
| Meaning: | Defines the number of predefined user frames. Approximately 400 bytes of backup memory are reserved per frame.<br>The standard configuration on the system provides four frames for G54 to G57 and one frame for G500. | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | |

<table>
<tr><td>28085<br>MD number</td><td colspan="3">MM_LINK_TOA_UNIT<br>Allocation of a TO unit to a channel</td></tr>
<tr><td>Default setting:<br>1, 2, 3, 4, 5 ...</td><td>Minimum input limit:<br>1</td><td colspan="2">Maximum input limit:<br>Max. no. chan. in system –1</td></tr>
<tr><td colspan="2">Changes effective after POWER ON</td><td>Protection level:</td><td>Unit: –</td></tr>
<tr><td colspan="2">Data type: DWORD</td><td colspan="2">Applies from SW: 2</td></tr>
<tr><td>Meaning:</td><td colspan="3">A TO unit is assigned to each channel through a default setting. The memory is thus reserved for the data blocks (tools, magazines).<br>A TOA unit can also be assigned to several channels.<br>Def.: The TOA area is the sum of all TOA and magazine blocks in the NC.<br>The TOA unit consists of a TOA block and, with activated TM function, a magazine block.</td></tr>
<tr><td>Special cases, errors, ......</td><td colspan="3">The buffered data are lost if this machine data is altered!</td></tr>
</table>

<table>
<tr><td>28090<br>MD number</td><td colspan="3">MM_NUM_CC_BLOCK_ELEMENTS<br>Number of block elements for Compile cycles</td></tr>
<tr><td>Default setting:<br>Compile cycles: 0<br>NC570: –<br>840di: 2, 2, 0, 0...<br>Transf. 810D: 2, 2, 0, 0 ...</td><td>Minimum input limit:<br>0<br>–<br>–<br>–</td><td colspan="2">Maximum input limit:<br>0<br>–<br>–<br>–</td></tr>
<tr><td colspan="2">Changes effective after POWER ON</td><td>Protection level: 0</td><td>Unit: –</td></tr>
<tr><td colspan="2">Data type: DWORD</td><td colspan="2">Applies from SW: 1.1</td></tr>
<tr><td>Meaning:</td><td colspan="3">The value defines the number of block elements used for compile cycles.<br>Approximately 1.2 KB of dynamic memory per block element is required for SW 2.</td></tr>
</table>

<table>
<tr><td>28100<br>MD number</td><td colspan="3">MM_NUM_CC_BLOCK_USER_MEM<br>Size of block memory for Compile cycles</td></tr>
<tr><td>Default setting:<br>NCU 570: –<br>NCU 572, 573: 256, 0, 0, 0...<br>840Di: 2, 2, 0, 0,<br>810D: 2, 2, 0, 0, 0</td><td>Minimum input limit:<br>–<br>–<br>–<br>–</td><td colspan="2">Maximum input limit:<br>–<br>–<br>–<br>–</td></tr>
<tr><td colspan="2">Changes effective after POWER ON</td><td>Protection level: 0</td><td>Unit: KB</td></tr>
<tr><td colspan="2">Data type: DWORD</td><td colspan="2">Applies from SW: 1.1</td></tr>
<tr><td>Meaning:</td><td colspan="3">The value defines the total capacity of block memory available to the user in the dynamic memory area for the compile cycles. The memory is allocated in staggered blocks of 128 bytes.</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| **28200**<br>MD number | **MM_NUM_PROTECT_AREA_CHAN**<br>Number of modules for channel-specific protection zones | | | | |
| Default setting:<br>0 | | Minimum input limit:<br>0 | | Maximum input limit:<br>10 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: |
| Data type: DWORD | | | | Applies from SW: 2 | |
| Meaning: | This machine data defines how many blocks are created for channel-specific protection zones. | | | | |
| Related to .... | MD 28210: MM_NUM_PROTECT_AREA_ACTIVE (number of simultaneously active protection zones)<br>MD 18190: MM_NUM_PROTECT_AREA_NCK (number of files for machine-related protection zones (SRAM)) | | | | |
| References | /FB/, A3, “Axis Monitoring, Protection Zones” | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **28210**<br>MD number | **MM_NUM_PROTECT_AREA_ACTIVE**<br>Number of simultaneously active protection zones | | | | |
| Default setting:<br>0 | | Minimum input limit:<br>0 | | Maximum input limit:<br>10 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 2 | |
| Meaning: | The machine data specifies for each channel the number of protection zones that may be activated simultaneously. The NCU-specific max. input limit specified above cannot be exceeded in total for all channel-specific parameters.<br><br>A numerical value higher than the setting in MD 18190: MM_NUM_PROTECT_AREA_NCK + MD 28200: MM_NUM_PROTECT_AREA_CHAN is not meaningful. | | | | |
| Related to .... | MD 28200: MM_NUM_PROTECT_AREA_CHAN (number of blocks for channel-specific protection zones)<br>MD 281212: MM_NUM_PROTECT_AREA_CONTOUR (elements for active protection zones (DRAM))<br>MD 18190: MM_NUM_PROTECT_AREA_NCK (number of files for machine-related protection zones (SRAM)) | | | | |
| References | /FB/, A3, “Axis Monitoring, Protection Zones” | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **28212**<br>MD number | **MM_NUM_PROTECT_AREA_CONTOUR**<br>Elements for active protection zones (DRAM) | | | | |
| Default setting:<br>30 | | Minimum input limit:<br>0 | | Maximum input limit:<br>50 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 6.4 | |
| Meaning: | This machine data specifies for each channel the number of internal contour elements to be kept available simultaneously for the individual active protection zones.<br><br>Dynamic memory space is used, determining the amount of memory required for active protection zones. This machine data only takes effect when MD 28210: MM_NUM_PROTECT_AREA_ACTIVE does not equal zero. | | | | |
| Related to .... | MD 28210: MM_NUM_PROTECT_AREA_ACTIVE (number of simultaneously active protection zones) | | | | |
| References | /FB/, A3, “Axis Monitoring, Protection Zones” | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **28500**<br>MD number | **MM_PREP_TASK_STACK_SIZE**<br>Stack size for preparation task | | | | |
| Default setting:<br>45 | | Minimum input limit:<br>10 | | Maximum input limit:<br>70 | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: KB |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | Defines the stack size for the preparation task. The stack is stored in dynamic memory. | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **28510**<br>MD number | **MM_IPO_TASK_STACK_SIZE**<br>Stack size of IPO task | | | | |
| Default setting:<br>NCU 571: 12<br>NCU 572: 12<br>NCU 573: 12<br>810D : 12 | | Minimum input limit:<br>NCU 571: 4<br>NCU 572: 4<br>NCU 573: 4<br>810D : 4 | | Maximum input limit:<br>NCU 571: 40<br>NCU 572: 40<br>NCU 573: 40<br>810D : 40 | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: KB |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | The stack size for the IPO task is stored in this data. The stack is set up in dynamic memory. | | | | |
| Note | This machine data is assigned internally by the control and must not be altered by the user. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **28550**<br>MD number | **MM_PRSATZ_MEM_SIZE**<br>Available memory for internal blocks | | | | |
| Default setting: 400 | | Minimum input limit: 100 | | Maximum input limit: 4000 | |
| Changes effective after POWER ON | | | Protection level: 0 | | Unit: KB |
| Data type: DWORD | | | | Applies from SW: 1.1 | |
| Meaning: | None.<br>This MD no longer exists in SW 2. | | | | |

## 4.3 Axis-specific machine data

| | | | | | |
|---|---|---|---|---|---|
| **38000**<br>MD number | **MM_ENC_COMP_MAX_POINTS**<br>Number of intermediate points with interpolatory compensation | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 5000 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 1,1 | |
| Meaning: | Defines the number of leadscrew compensation values per encoder for the axis. This value reserves 8 bytes of backup user memory for each compensation value. If more memory for compensation values is required than available in the SRAM, the control outputs alarm 6000 “Memory allocation with standard machine data” on power-up. | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | |
| References | /FB/, K3, “Compensation” | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **38010**<br>MD number | **MM_QEC_MAX_POINTS**<br>Number of values for quadrant error compensation | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1040 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 2 | |
| Meaning: | Number of possible values for quadrant error compensation with neural network (option).<br>When value = 0 : The quadrant error compensation function cannot be activated, no memory is set up for the function. | | | | |
| Special cases, errors, ...... | The buffered data are lost if this machine data is altered! | | | | |
| References | /IAD/, “SINUMERIK 840D Installation and Start-up Guide”<br>/FB/, K3, “Compensations” | | | | |

■

# Notes

# 5 Signal Descriptions

None

# 6 Example

None

# 7 Data Fields, Lists

## 7.1 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| General ($MN_ ...) | | | |
| 18050 | INFO_FREE_MEM_DYNAMIC | Display data of the free dynamic memory | |
| 18060 | INFO_FREE_MEM_STATIC | Display data of the free static memory | |
| 18070 | INFO_FREE_MEM_DPR | Display data of free memory in DUAL_PORT RAM | |
| 18080 | MM_TOOL_MANAGEMENT_MASK | Screen form for reserving memory for the tool management | /FBW/ |
| 18082 | MM_NUM_TOOL | Number of tools managed by NCK | |
| 18084 | MM_NUM_MAGAZINE | Number of magazines managed by NCK | /FBW/ |
| 18086 | MM_NUM_MAGAZINE_LOCATION | Number of magazine locations | /FBW/ |
| 18090 | MM_NUM_CC_MAGAZINE_PARAM | Compile cycles of tool management: Number of magazine data | /FBW/ |

| Number | Names | Name | Reference |
|---|---|---|---|
| General ($MN_ ...) | | | |
| 18092 | MM_NUM_CC_MAGLOC_PARAM | Compile cycles of tool management: Number of magazine location data | /FBW/ |
| 18094 | MM_NUM_CC_TDA_PARAM | Compile cycles of tool management: Number of TDA data | /FBW/ |
| 18096 | MM_NUM_CC_TOA_PARAM | Compile cycles of tool management: Number of TOA data | /FBW/ |
| 18098 | MM_NUM_CC_MON_PARAM | Compile cycles of tool management: Number of monitor data | /FBW/ |
| 18100 | MM_NUM_CUTTING_EDGES_IN_TOA | Number of tool offsets in NCK | |
| 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_GUNAMES_AXIS | No. of axis-specific user variables | |
| 18150 | MM_GUD_VALUES_MEM | Memory reserved 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 | /FB/, A3 |
| 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 | |
| 18240 | MM_LUD_HASH_TABLE_SIZE | Hash table size for user variables | |
| 18242 | MM_MAX_SIZE_OF_LUD_VALUE | Maximum field size of the LUD variables | |
| 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 | |
| 18320 | MM_NUM_FILES_IN_FILESYSTEM | Number of files in passive file system | |
| 18330 | MM_CHAR_LENGTH_OF_BLOCK | Max. length of an NC block | |
| 18340 | MM_NUM_CEC_NAMES | Number of LEC tables | |
| 18342 | MM_CEC_MAX_POINTS | Maximum table size for sag compensation | |
| 18350 | MM_USER_FILE_MEM_MINIMUM | Minimum NC program memory | |
| 18500 | MM_EXTCOM_TASK_STACK_SIZE | Stack size for external communication task | |
| 18510 | MM_SERVO_TASK_STACK_SIZE | Stack size of servo task | |
| 18520 | MM_DRIVE_TASK_STACK_SIZE | Stack size of drive task | |
| Channelspecific ($MC_ ...) | | | |

| Number | Names | Name | Reference |
|---|---|---|---|
| General ($MN_ ...) | | | |
| 20096 | T_M_ADDRESS_EXIT_SPINO | Spindle number as address extension (SW 5 and higher) | /FBW/, W1 |
| 27900 | REORG_LOG_LIMIT | Percentage of IPO buffer for log file enable | |
| 28000 | MM_REORG_LOG_FILE_MEM | Memory size for REORG | /FB/, K1 |
| 28010 | MM_NUM_REORG_LUD_MODULES | Number of modules for local user variables with REORG | |
| 28020 | MM_NUM_LUD_NAMES_TOTAL | Number of local user variables | |
| 28040 | MM_LUD_VALUES_MEM | Memory size for local user variables | |
| 28050 | MM_NUM_R_PARAM | Number of channel-specific R parameters | |
| 28060 | MM_IPO_BUFFER_SIZE | Number of NC blocks in the IPO buffer | |
| 28070 | MM_NUM_BLOCKS_IN_PREP | Number of blocks for block preparation | |
| 28080 | MM_NUM_USER_FRAMES | Number of settable frames | |
| 28085 | MM_LINK_TOA_UNIT | Allocation of a TO unit to a channel | /FBW/, W1 |
| 28090 | MM_NUM_CC_BLOCK_ELEMENTS | Number of block elements for Compile cycles | |
| 28100 | MM_NUM_CC_BLOCK_USER_MEM | Size of block memory for Compile cycles | |
| 28200 | MM_NUM_PROTECT_AREA_CHAN | Number of modules for channel-specific protection zones | /FB/, A3 |
| 28210 | MM_NUM_PROTECT_AREA_ACTIVE | Number of simultaneously active protection zones | /FB/, A3 |
| 28212 | MM_NUM_PROTECT_AREA_CONTOUR | Elements for active protection zones (DRAM) | /FB/, A3 |
| 28500 | MM_PREP_TASK_STACK_SIZE | Stack size of preparation task | |
| 28510 | MM_IPO_TASK_STACK_SIZE | Stack size of IPO task | |
| 28550 | MM_PRSATZ_MEM_SIZE | Available memory for internal blocks | |
| Axisspecific ($MA_ ...) | | | |
| 38000 | MM_ENC_COMP_MAX_POINTS | Number of intermediate points with interpolatory compensation | /FB/, K3 |
| 38010 | MM_QEC_MAX_POINTS | Number of values for quadrant error compensation | /FB/, K3 /IAD/ |

## 7.2 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, “Diagnostics Guide”
and the online help of MMC 101/102/103 systems.

■

# Notes

# SINUMERIK 840D/840Di/810D Description of Functions Extended Functions (FB2)

# Indexing Axes (T1)



■

## Notes

# 1 Brief Description

**Indexing axes on 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 (called indexes) both in automatic and set-up modes. These axes are known as “indexing axes”. The positions defined on the indexing axes are known as “coded positions” or “index positions”.
Special functions are available for equidistant indexing on linear and rotary axes and for the Hirth tooth system.

**Applications**

Indexing axes are used predominantly in connection with specific types of tool magazine 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 index**

A system variable can specify the number of the current indexing position depending on the specifications in a machine data:

– when the exact stop fine window of the index position is reached or

– when half the distance to the next indexing position is crossed.

The programmed index position can be scanned using a further system variable.

■

# Notes

# Detailed Description 2

## 2.1 Traversing indexing axes

**General**

Indexing axes can be traversed manually in the setup mode types JOG and INC, from a parts program with special instructions for “Coded positions” and by the PLC. When the indexing position has been reached, the “indexing axis in position” interface signal (DB31-61, DBX76.6) is output to the PLC.

Hirth indexing axes cannot be traversed in JOG mode before reference point approach.

### 2.1.1 Traversing indexing axes in manual JOG mode

**Reference point approach**

An indexing axis approaches the reference point in the same way as other axes. It is not necessary for the reference point to match an indexing position. Only **when the reference point has been reached** (IS “Referenced/synchronized 1 or 2” (DB31-61, DBX60.4 or 5) = “1”) does the indexing axis start to approach indexing positions in JOG mode with JOG and incremental traversing. Exception: When traversing with the handwheel, no indexing positions are approached.

If the axis is not referenced (IS “Referenced/synchronized 1 or 2” = “0”), the indexing positions are ignored when the axis is traversed in manual jog mode!

**Continuous traversal in JOG**

- Jog mode (SD: 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 (SD: 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, please see:
**References:** /FB/, H1 “Manual and Handwheel Travel”

**Incremental jogging (JOG INC)**

Irrespective of the current increment setting (INC1; ... ,INCvar), the indexing axis **is always incremented by 1 indexing position** in the selected direction when a “+” or “–” traversing key 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 aborted 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**. Rotating the handwheel traverses the indexing axis to any position in mm, inches or degrees, according to the selected unit of measurement.
The PLC user program can disable the handwheel for traversing the indexing axis.

**Signal from PLC “Indexing axis in position”**

When the indexing axis is traversed in JOG mode, the signal “Indexing axis in position” (DB31-61, DBX76.6) is output at the PLC interface to indicate that an indexing position has been reached. The indexing axis must have been referenced (IS “Referenced/synchronized 1 or 2” = “1”).

**Alarms in JOG mode**

If the indexing axis leaves the traversing range defined in the indexing position table (see 2.2) when traversing in JOG mode, alarm 20054 “wrong index for indexing axis in JOG” is output.

**Revolutional feedrate**

In JOG mode the behavior of the axis/spindle also depends on the setting of setting data JOG_REV_IS_ACTIVE (revolutional feedrate when JOG active).

- If this setting data is active, an axis/spindle is always moved with revolutional feedrate MD JOG_REV_VELO (revolutional feedrate with JOG) or MD JOG_REV_VELO_RAPID (revolutional feedrate with JOG with rapid traverse overlay) depending on the master spindle.
- If the setting data is not active, the behavior of the axis/spindle depends on the setting data ASSIGN_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 JOG_FEED_PER_REV_SOURCE. (In the operating mode JOG, revolutional feedrate for geometry axes on which a frame with rotation is effective).

## 2.1.2 Traversing indexing axes in AUTOMATIC modes

**Traversal to selected positions**

An axis defined as an indexing axis can be made to approach **any selected position** from the NC parts 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 Guide.

**Traversal to “Coded positions”**

Special instructions can also be programmed in the parts program:

- 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

to traverse in the specified manner.

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). Please refer to Section 2.3 for further information on the special programming instructions for indexing axes.

**Interface signal "Indexing axis in position"**

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.

### 2.1.3 Traversing of indexing axes by PLC

**Traversal from PLC**

Indexing axes can also be traversed from the PLC user program. There are various methods:

- With concurrent positioning axes
  In this case, the indexing position to be approached can be specified by the PLC.
  **References:** /FB/, P2, "Positioning Axes"
- With asynchronous subprograms (ASUB)
  **References:** /FB/, K1 "Mode Groups, Channels, Program Operation"

## 2.2 Parameterization of indexing axes

**Definition of the indexing axis**

An axis (linear or rotary axis) can be declared as an indexing axis using machine data MD 30500: INDEX_AX_ASSIGN_POS_TAB. The number of the indexing position table (1 or 2) must be entered in the machine data.

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). Otherwise alarm 4080 "Incorrect configuration for indexing axis in MD [Name]" is output at boot.

**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. Any value can be entered for the distance between the individual indexing positions.
The following should be noted when entering the indexing positions:

**Number of tables**

Up to two indexing position tables are permitted:

MD 10910: INDEX_AX_POS_TAB_1 [n]
MD 10930: INDEX_AX_POS_TAB_2 [n]

**Number of entries for each table**

Up to 60 positions can be entered in each indexing position table [n = 0 ... 59].

The actual number of entries used must be defined with machine data

MD 10900: INDEX_AX_LENGTH_POS_TAB_1 or
MD 10920: INDEX_AX_LENGTH_POS_TAB_2

for table 1 and/or 2.

All positions entered in the table which are higher than the number defined in the machine data are inactive.

**Inch/metric switchover**

The indexing positions refer to the measurement system defined with MD 10270: POS_TAB_SCALING_SYSTEM.

MD 10270: POS_TAB_SCALING_SYSTEM=0: metric
MD 10270: POS_TAB_SCALING_SYSTEM=1: Inch

---

**Note**

MD 10270 defines the system of units for position specifications for the following machine data:
MD 10900: INDEX_AX_POS_TAB_1
MD 10920: INDEX_AX_POS_TAB_2

MD 10270 also affects SD 41500 to SD 41507 (see /N3/).

---

**Entry format**

- The indexing positions should 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 should be entered in the basic coordinate system.

If the indexing axis is defined as a rotary axis with modulo 360º (MD: IS_ROT_AX = "1" and MD: ROT_IS_MODULO = "1"), the following points should also be observed with respect to indexing positions:

- Indexing positions may be programmed in the range from 0º ≦ Pos < 360º. Positions outside this range generate alarm 4080 on power-up.
- 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.

## 2.3 Programming of indexing axes

**Note**

Detailed information about programming indexing axes can be found in
**References:** /PA/, "Programming Guide: Fundamentals"

**Coded position**

To allow indexing axes to be positioned from the NC parts program, special instructions (so-called **C**oded positions) are provided with which the indexing numbers (e.g. location number) are programmed rather than axis positions in mm or degrees. The following coded position instructions are possible, depending on whether the indexing axis is defined as a linear or rotary axis:

Indexing axis is a linear axis: CAC(i), CIC(i)
Indexing axis is a rotary axis: CAC(i), CIC(i), CACP(i), CACN(i), CDC(i)

i = coded position (indexing position)
Value range of i: 0 ... 59; integer; with the exception of CIC positive only.

**Absolute**
**POS[B]=CAC(20)**

Indexing axis B approaches **c**oded position (index) 20 in absolute mode. The direction of traversing depends on the current actual position.

**Absolute in positive direction**
**POS[B]=CACP(10)**

Indexing axis B approaches **c**oded position (index) 10 in absolute mode in the positive direction of rotation (only possible with rotary axes).

**Absolute in negative direction**
**POS[B]=CACN(10)**

Indexing axis B approaches **c**oded position (index) 10 in absolute mode in the negative direction of rotation (only possible with rotary axes).

**Direct absolute**
**POS[B]=CDC(50)**

Indexing axis B approaches indexing position 50 directly along the shortest path (only possible with rotary axes).

**Incremental**
**POS[B]=CIC(-4)**

Indexing axis B traverses incrementally by four indexing positions in a negative direction from its current position.

**POS[B]=CIC(35)**

Indexing axis B traverses incrementally from the current indexing position across 35 indexing positions in the positive direction.
The leading sign defines the direction of approach.

---

**Note**

On modulo rotary axes, the indexing positions are divided in factors of 360 degrees and are approached directly.

---

**Between indexing positions**

If an indexing axis is located between two indexing positions in automatic mode, the program command

POS[B]=CIC(1) causes the next higher indexing position to be approached.
Similarly, the program instruction
POS[B]=CIC(–1) causes the next lower indexing position to be approached.
With
POS[B]=CIC(0) the indexing axis does not traverse.

**Display of indexing position**

The last **programmed** indexing position can be read with system variable **$AA_PROG_INDEX_AX_POS**.

The system variable for the number of the last indexing position crossed **$AA_ACT_INDEX_AX_POS** can indicate the following depending on the setting in MD 10940: INDEX_AX_MODE:

MD 10940: $MN_INDEX_AX_MODE **Bit0 = 0**
(behavior compatible with older software versions) means:
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 immediately before the next indexing position.

MD 10940: $MN_INDEX_AX_MODE **Bit0 = 1**
means:
The indexing position changes when **half** the indexing position is reached. A quasi-symmetrical indexing area thus applies around the indexing position (symmetrical only in the case of linear axes with equidistant partitioning or modulo rotary axes on which the indexing area is an integer multiple of the modulo range (MD 30330: MODULO_RANGE), otherwise **proportional to the distances** between the indexing positions).
With **modulo rotary axes**, the area between the last indexing position and the first indexing position is distributed **proportionally** based on the lengths of the first and last indexing area.

The following figure illustrates the difference for Bit0 = 0 and Bit0 = 1:

Legend:

| | |
|---|---|
| TP —> | Programmed indexing position |
| TPA—> | Displayed indexing position |
| GHFF—> | Exact stop fine window |

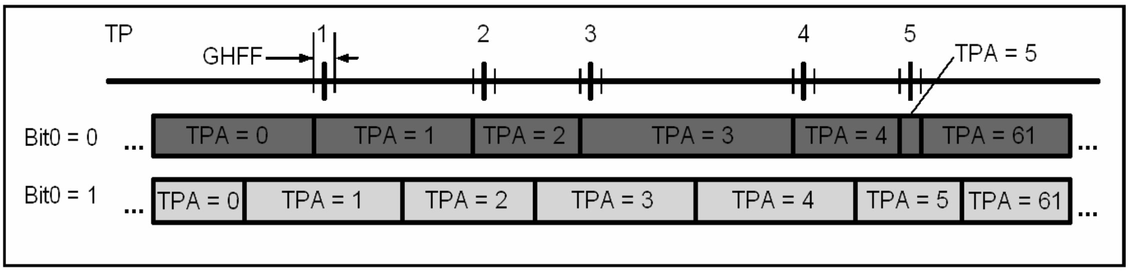



Fig. 2-1 Indexing position displays, linear axis (tabular indexing axis positions)

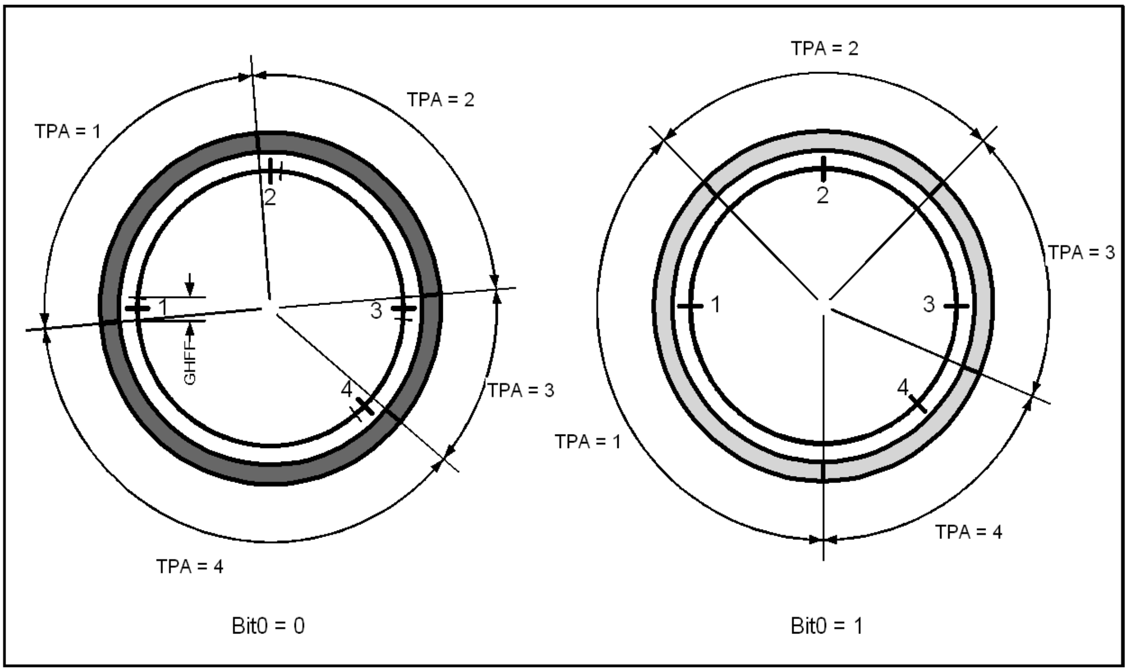



Fig. 2-2 Indexing position displays, modulo rotary axis (tabular indexing positions)

**$AA_ACT_ INDEX_ AX_POS_NO**

The anticipated value ranges of the system variable **$AA_ACT_INDEX_AX_POS** (OPI variable aaActIndexAxPosNo) are:

Table 2-1 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,<br>61: bove total indexing area |

Table 2-2 Equidistant indexing positions

| Modulo rotary axis | 1 . . . m | None 0,<br>m = denominator |
|---|---|---|
| Linear axis | 0 1 2 3 . . . 65535 | actIndexAxPosNo (old variable) |
| Linear axis | ... –3 –2 –1 0 1 2 3 . . . | **aa**ActIndexAxPosNo |

**Note**

OPT variable **actIndexAxPosNo** exists for compatibility reasons only. Only OPI variable **aaActIndexAxPosNo** should be used where possible.

### Next indexing position

Behavior with command **"Traverse to next indexing position"**
Bit0 = 0: Next indexing position is approached
Bit0 = 1: The next indexing position in the direction of travel is always approached

Explanation:
Bit0 = 1 and the axis is below the indexing position (but outside the exact stop fine window):
Although, e.g. $AA_ACT_INDEX_AX_POS_NO indexing position 2 is displayed, indexing position 2 and NOT indexing position 3 is approached initially. The next indexing position (indexing position 3 in the example) is not approached with "Traverse to next position" until the axis is positioned exactly on (exact stop fine) or above the indexing position.
The next indexing position in the direction of travel is always approached first! It is therefore sometimes necessary to dispatch the "Traverse to next position" command twice in order to proceed from the currently displayed indexing position to the next indexing position number (e.g. from 2 to 3).

**Alarms**

If an indexing position is programmed outside the valid range of the indexing position table, alarm 17510 “Impermissible index for indexing axis” is output. When an indexing position is programmed for an axis, alarm 17500 “Axis is not an indexing axis” is generated if an indexing position table is not assigned to this axis (MD: INDEX_AX_ASSIGN_POS_TAB (axis is an indexing axis)).

**FRAMES**

Since the control interprets the positions stored in the indexing position table as programmed positions in mm, inches or degrees, FRAMES are not disabled with indexing axes.
FRAMES are not generally required with indexing axes, depending on the application. It is therefore in most cases advisable to suppress FRAMES and zero offsets in the parts program for indexing axes.

## 2.4 Equidistant index intervals

### 2.4.1 Function

**General**

The following apply:

- Any number of equidistant index intervals
- Modified action of MD for indexing axes

Equidistant index intervals can be used for:

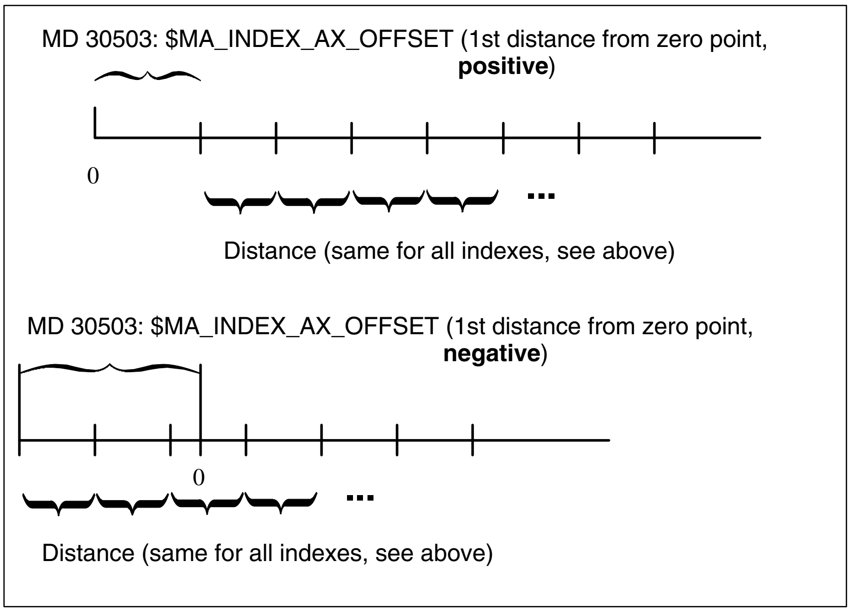
- Linear axes
- Modulo rotary axes
- Rotary axes

**Distance between indexes**

The index distance is determined as follows for equidistant index intervals:

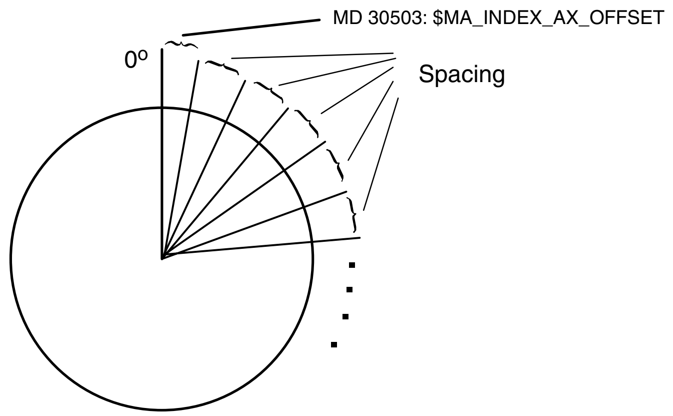
$$\text{Distance} = \frac{\text{Numerator (MD 30501: \$MA\_INDEX\_AX\_NUMERATOR)}}{\text{Denominator (MD 30502: \$MA\_INDEX\_AX\_DENOMINATOR)}}$$

**Linear axis**



**Modulo rotary axis**

$$\text{Index} = \frac{\text{Numerator (MD 30330: \$MA\_MODULO\_RANGE)}}{\text{Denominator (MD 30502: \$MA\_INDEX\_AX\_DENOMINATOR)}}$$




$$\text{Index} = \frac{\text{Size of modulo range (MD 30330: \$MA\_MODULO\_RANGE)}}{\text{Number of indexes in modulo range (MD 30502: \$MA\_INDEX\_AX\_DENOMINATOR)}}$$

**Activation**

The functions with equidistant indexing for linear axes and rotary axes or modulo rotary axes are activated by specifying "table number" 3 in MD 30500: $MA_INDEX_AX_ASSIGN_POS_TAB[axis].

## Hirth tooth system

**Introduction**

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.

**Prerequisites**

The rotary axis must be an indexing axis. The axis must be referenced. See
**References:** /R1/, Reference Point Approach

**Activation**

MD 30505: $MA_HIRTH_IS_ACTIVE must be set to 1.
MD 30500: $MA_INDEX_AX_ASSIGN_POS_TAB must be set to 3 (equidistant indexes).

**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.
  Precondition:
  The axis is referenced.
- Jogging with the handwheel is not possible. see
  **References:** /H1/, Handwheel Travel
- Only "coded positions" can be approached in AUTO, MDA or via ASUBs
- The PLC can only move the axis to indexing positions. An alarm is output on an attempt to approach any other position.

## Response of the Hirth axes in particular situations

**STOP/RESET**

On NC STOP and RESET during a traversing movement, the next indexing position is approached before the command is activated.

**EMERGENCY STOP**

After an EMERGENCY STOP, the PLC or the operator must move the indexing axis back to an indexing position in JOG mode before the longitudinal axis can be moved in/down.

**Override = 0 or "Stop axis" signal**

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.

**Deletion of distance-to-go**

After traversing to the next possible indexing position, the movement is aborted at this position.

**Command axes**

See **References:**/FBSY/, Synchronized Actions
If MOV = 0 is specified for a moving command axis, the axis continues traversing to the next possible indexing position.

**Move command**

MOV = 1 Works on indexing axes with and without Hirth tooth system. Move = 0 works the same with both, the next position is approached.

**DELDTG command**

For indexing axes without Hirth tooth system:axis stops immediately...
For indexing axes with Hirth tooth system:axis approaches next position.

## 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.

**Rev. 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, external WO, DRF**

The axis for which the Hirth tooth system is defined does not support frames or interpolation compensation such as external zero offsets, DRF, etc.

**Couplings**

The axis for which the Hirth tooth system is defined cannot be a

- following axis with master value coupling
- coupled-motion axis
- gantry following axis.

See:
**References:** /M3/, Coupled Motion

### 2.4.2 Modified activation of machine data

A RESET is required in order to activate the MD below after new values have been assigned to them.

MD 10900: $MN_INDEX_AX_LENGTH_POS_TAB_1

MD 10920: $MN_INDEX_AX_LENGTH_POS_TAB_2

MD 10910: $MN_INDEX_AX_POS_TAB_1

MD 10930: $MN_INDEX_AX_POS_TAB_2

MD 30500: $MA_INDEX_AX_ASSIGN_POS_TAB

You will find a complete list of MD for indexing axes in Chapter 4.

### 2.4.3 Examples of equidistant indexes

**Modulo rotary axis**

$MA__INDEX_AX_DENOMINATOR[AX4] =18
$MA_INDEX_AX_OFFSET[AX4]=5
$MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3
$MA_IS_ROT_AX[AX4] = TRUE
$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 degrees starting at 5 degrees. This results in the following indexing positions: 5, 25, 45, 65, 85, 105, 125, 145, 165, 185, 205, 225, 245, 265, 285, 305, 325 and 245 degrees.

---

**Note**

The $MA__INDEX_AX_DENOMINATOR[AX4] =18 assignment produces a 20° division because the default for $MA_MODULO_RANGE is 360.

---

**Rotary axis**

$MA_INDEX_AX__NUMERATOR[AX4] = 360
$MA__INDEX_AX_DENOMINATOR[AX4] =18
$MA_INDEX_AX_OFFSET[AX4]=100
$MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3
$MA_IS_ROT_AX[AX4] = TRUE
$MA_POS_LIMIT_MINUS[AX1]=100
$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 degrees starting at 100 degrees. This results in the following indexing positions: 100, 120, 140 degrees etc. Positions less than 100 degrees cannot be approached as indexing positions. It is therefore 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 degrees). The rotary axis can therefore only traverse between 100 and 260 degrees.

**Linear axis**

$MA_INDEX_AX__NUMERATOR[AX1] = 10
$MA__INDEX_AX_DENOMINATOR[AX1] =1
$MA_INDEX_AX_OFFSET[AX1]=–200
$MA_INDEX_AX_ASSIGN_POS_TAB[AX1] = 3
$MA_IS_ROT_AX[AX1] = FALSE
$MA_POS_LIMIT_MINUS[AX1]=–200
$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. This results in the following indexing positions: –200, –190, –180 mm etc. The indexing positions continue until the software limit switch is reached (in this case 200 mm).

**Hirth tooth system**

$MA__INDEX_AX_DENOMINATOR[AX4] =360
$MA_INDEX_AX_OFFSET[AX4]=0
$MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3
$MA_IS_ROT_AX[AX4] = TRUE
$MA_ROT_IS_MODULO[AX5] = TRUE
$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 degree starting at 0 degrees.

## 2.5 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 (MD: IS_ROT_AX = "1") with modulo 360° conversion (MD: ROT_IS_MODULO = "1"), the indexing positions are traversed 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. MD: DISPLAY_IS_MODULO = "1" can be programmed to set the position display to modulo 360°.

**Special machine data**

The following machine data, described in Chapter 4, must also be defined:

**General machine data**

| | |
|---|---|
| MD: INDEX_AX_LENGTH_POS_TAB_1 | (no. of indexing positions used in table 1) |
| MD: INDEX_AX_LENGTH_POS_TAB_2 | (no. of indexing positions used in table 2) |
| MD: INDEX_AX_POS_TAB_1 [n] | Indexing position table 1 |
| MD: INDEX_AX_POS_TAB_2 [n] | Indexing position table 2 |

**Axial machine data**

| | |
|---|---|
| MD: INDEX_AX_ASSIGN_POS_TAB | Axis is an indexing axis (assignment of indexing position table 1 or 2, or 3 for equidistant indexing) |
| MD: HIRTH_IS_ACTIVE | Axis has "Hirth tooth system" property, |
| MD: INDEX_AX_NUMERATOR | Numerator for equidistant indexing |
| MD: INDEX_AX_DENOMINATOR | Denominator for equidistant index ing |
| MD: INDEX_AX_OFFSET | Distance of the 1st indexing position from zero |

**Machine data examples**

The assignment of the above machine data is described in the following paragraphs using two examples.

**Example of indexing axis as rotary axis**

Tool turret with eight turret locations

The tool turret is defined as a continuously rotating rotary axis. The distances between the eight turret locations are constant, the first location is at position 0° (see Fig. 2-3).

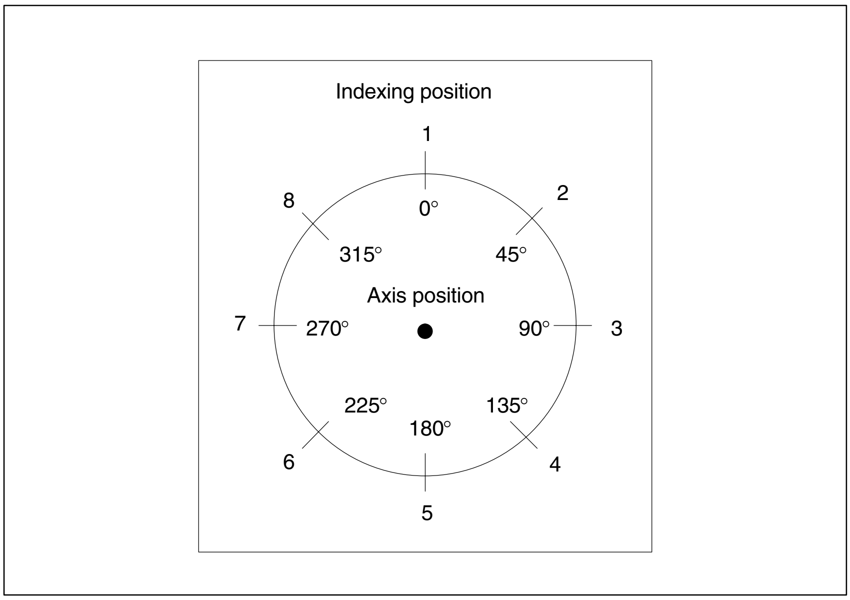



Fig. 2-3 Example: Tool turret with 8 locations

**Indexing position table**

The indexing positions for the tool turret are entered in table 1.

```
$MN_INDEX_AX_POS_TAB_1[0] = 0        ; 1st indexing position at 0°
$MN_INDEX_AX_POS_TAB_1[1] = 45       ; 2nd indexing position at 45°
$MN_INDEX_AX_POS_TAB_1[2] = 90       ; 3rd indexing position at 90°
$MN_INDEX_AX_POS_TAB_1[3] = 135      ; 4th indexing position at 135°
$MN_INDEX_AX_POS_TAB_1[4] = 180      ; 5th indexing position at 180°
$MN_INDEX_AX_POS_TAB_1[5] = 225      ; 6th indexing position at 225°
$MN_INDEX_AX_POS_TAB_1[6] = 270      ; 7th indexing position at 270°
$MN_INDEX_AX_POS_TAB_1[7] = 315      ; 8th indexing position at 315°
```

**Other machine data**

```
$MN_INDEX_AX_LENGTH_POS_TAB_1= 8     ; 8 indexing positions in
                                     Table 1
$MA_INDEX_AX_ASSIGN_POS_TAB [AX5] = 1; Axis 5 is defined as an index
                                       ing axis, indexing positions
                                       in Table 1
$MA_IS_ROT_AX [AX5] = 1              ; Axis 5 is a rotary axis
$MA_ ROT_IS_MODULO [AX5] = 1         ; Modulo conversion is
                                       activated
```

**Example of indexing axis as linear axis**

Workholder with ten locations (see Fig. 2-4).
The distances between the ten locations vary; the first workholder location is at position –100 mm.

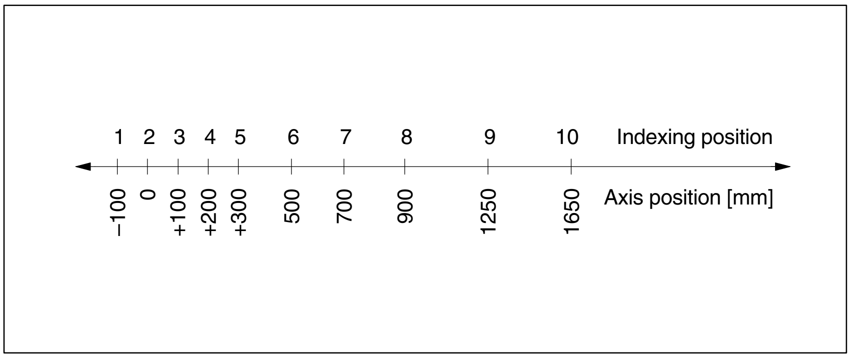



Fig. 2-4 Example: Workholding pallet as an indexing axis

**Indexing position table**

The indexing positions for the tool turret are entered in table 2.

```
$MN_INDEX_AX_POS_TAB_2[0] = –100     ; 1st indexing position at –100
$MN_INDEX_AX_POS_TAB_2[1] = 0        ; 2nd indexing position at 0
$MN_INDEX_AX_POS_TAB_2[2] = 100      ; 3rd indexing position at 100
$MN_INDEX_AX_POS_TAB_2[3] = 200      ; 4th indexing position at 200
$MN_INDEX_AX_POS_TAB_2[4] = 300      ; 5th indexing position at 300
$MN_INDEX_AX_POS_TAB_2[5] = 500      ; 6th indexing position at 500
$MN_INDEX_AX_POS_TAB_2[6] = 700      ; 7th indexing position at 700
$MN_INDEX_AX_POS_TAB_2[7] = 900      ; 8th indexing position at 900
$MN_INDEX_AX_POS_TAB_2[8] = 1250     ; 9th indexing position at 1250
$MN_INDEX_AX_POS_TAB_2[9] = 1650     ; 10th indexing position at 1650
```

**Other machine data**

```
$MN_INDEX_AX_LENGTH_POS_TAB_2=10   ; 10 indexing positions in
                                     Table 2
$MA_INDEX_AX_ASSIGN_POS_TAB [AX6] = 2 ; Axis 6 is defined as an index
                                     ing axis
                                   ; indexing positions in Table 2
```

## 2.6 Special features of indexing axes

**DRF**

An additional incremental zero offset can also be generated for indexing axes in AUTOMATIC mode with the handwheel using the DRF function.

**Software limit switches**

After the indexing axis has been referenced, the software limit switches are active when the axis is traversed.
When traversing manually in continuous JOG or incremental JOG mode, the indexing axis stops at the last indexing position before the software limit switch.

**Reference point approach**

An indexing axis will approach indexing positions in JOG mode (continuous or incremental) only **after it has reached its reference point** (IS "Referenced/ synchronized 1 or 2" (DB31-48, DBX60.4 or 5) = "1").
If the axis is not referenced (IS "referenced/synchronized 1 or 2" = "0"), the indexing positions are ignored when traversing manually!

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.

---

**Note**

The response of the Hirth tooth system is described in Subsection 2.4.1.

---

■

# Supplementary Conditions 3

There are no supplementary conditions stipulated for this Description of Functions.

■

# Data Descriptions (MD, SD) 4

## 4.1 General machine data

| | | | | | |
|---|---|---|---|---|---|
| **10270**<br>MD number | **POS_TAB_SCALING_SYSTEM**<br>Measuring system of the position tables | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after RESET | | | Protection level: 2 / 7 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 5 | |
| Meaning: | This machine data is for setting the measuring system for position specifications of indexing axis tables and switching points for software cams.<br>• MD 10270=0: Metric<br>• MD 10270=1: Inch<br>MD 10270 defines the measuring system for position specifications for the following machine data:<br>MD 10900: INDEX_AX_POS_TAB_1<br>MD 10920: INDEX_AX_POS_TAB_2<br>SD 41500: SW_CAM_MINUS_POS_TAB_1<br>SD 41501: SW_CAM_PLUS_POS_TAB_1<br>SD 41502: SW_CAM_MINUS_POS_TAB_2<br>SD 41503: SW_CAM_PLUS_POS_TAB_2<br>SD 41504: SW_CAM_MINUS_POS_TAB_3<br>SD 41505: SW_CAM_PLUS_POS_TAB_3<br>SD 41506: SW_CAM_MINUS_POS_TAB_4<br>SD 41507: SW_CAM_PLUS_POS_TAB_4<br><br>**Note: Only effective when MD 10260: CONVERT_SCALING_SYSTEM=1. (see /G2/)** | | | | |
| Related to .... | See machine and setting data under significance;<br>MD 10260: CONVERT_SCALING_SYSTEM | | | | |

| 10900<br>MD number | INDEX_AX_LENGTH_POS_TAB_1<br>Number of indexing positions used in Table 1 | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 60 |
| Modification effective after power ON or RESET with SW 4.3 and higher | Protection level: 2 / 7 | Unit: – |
| Data type: DWORD | Applies from SW: 1.1 | |
| Meaning: | The indexing position table is used to assign the axis positions in the valid unit of measurement (mm, inches or degrees) to the indexing positions [n] on the indexing axis.<br>The number of indexing positions used in table 1 is defined by the MD: INDEX_AX_LENGTH_POS_TAB_1.<br>These indexing positions must contain valid values in table 1. Any indexing positions in the table greater than the number specified in the machine data are ignored.<br>Up to 60 indexing positions (0 to 59) can be entered in the table.<br>Table length = 0 means that the table is not evaluated. If the length is not equal to 0, the table must be assigned to an axis with the MD: INDEX_AX_ ASSIGN_POS_TAB.<br><br>If the indexing axis is defined as a rotary axis (MD: IS_ROT_AX = "1") with modulo 360° (MD: ROT_IS_MODULO = "1"), the machine data defines the last indexing position after which the indexing positions begin again at 1 with a further traversing movement in the positive direction. | |
| Application example(s) | Tool magazines (tool turrets, chain magazines) | |
| Special cases, errors, ..... | Alarm 17090 "Value violates upper limit" if a value over 60 is entered in the MD: INDEX_AX_LENGTH_POS_TAB_1. | |
| Related to .... | MD: INDEX_AX_ASSIGN_POS_TAB<br>MD: INDEX_AX_POS_TAB_1<br>MD: IS_ROT_AX<br>MD: ROT_IS_MODULO | (axis is an indexing axis)<br>(indexing position table 1)<br>(rotary axis)<br>(modulo conversion for rotary axis) |

| 10920<br>MD number | INDEX_AX_LENGTH_POS_TAB_2<br>Number of indexing positions used in Table 2 | |
|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 60 |
| Modification effective after power ON or RESET with SW 4.3 and higher | Protection level: 2 / 7 | Unit: – |
| Data type: DWORD | Applies from SW: 1.1 | |
| Meaning: | The indexing position table is used to assign the axis positions in the valid unit of measurement (mm, inches or degrees) to the indexing positions [n] on the indexing axis.<br>The number of indexing positions used in table 2 is defined by the MD: INDEX_AX_LENGTH_POS_TAB_2.<br>These indexing positions must contain valid values in table 2. Any indexing positions in the table greater than the number specified in the machine data are ignored.<br>Up to 60 indexing positions (0 to 59) can be entered in the table.<br>Table length = 0 means that the table is not evaluated. If the length is not equal to 0, the table must be assigned to an axis with the MD: INDEX_AX_ASSIGN_POS_TAB.<br><br>If the indexing axis is defined as a rotary axis (MD: IS_ROT_AX = "1") with modulo 360° (MD: ROT_IS_MODULO = "1"), the machine data defines the last indexing position after which the indexing positions begin again at 1 with a further traversing movement in the positive direction. | |
| MD irrelevant for ... ... | Tool magazines (tool turrets, chain magazines) | |
| Special cases, errors, ... ... | Alarm 17090 "Value violates upper limit" if a value over 60 is entered in the MD: INDEX_AX_LENGTH_POS_TAB_2. | |
| Related to .... | MD: INDEX_AX_ASSIGN_POS_TAB<br>MD: INDEX_AX_POS_TAB_2<br>MD: IS_ROT_AX<br>MD: ROT_IS_MODULO | (axis is an indexing axis)<br>(indexing position table 2)<br>(rotary axis)<br>(modulo conversion for rotary axis) |

| | | | | | |
|---|---|---|---|---|---|
| **10910**<br>MD number | **INDEX_AX_POS_TAB_1[n]**<br>Indexing position table 1 [n] | | | | |
| Default setting: 0 | | Minimum input limit: *** | | Maximum input limit: *** | |
| Modification effective after power ON or RESET with SW 4.3 and higher | | | Protection level: 2 / 7 | | Unit: mm or degrees |
| Data type: DOUBLE | | | | Applies from SW: 1.1 | |
| Meaning: | The indexing position table is used to assign the axis positions in the valid unit of measurement (mm, inches or degrees) to the indexing positions [n] on the indexing axis.<br>[n] = Index for the entry of the indexing positions in the indexing position table.<br>Range: 0 $\geqq$ n $\leqq$ 59, where 0 is the 1st indexing position and 59 the 60th indexing position.<br>Note: Programming with the absolute indexing position (e.g. CAC) starts with indexing position 1. This corresponds to the indexing position with index n = 0 entered in the table of indexing positions.<br>The following should be noted when entering the indexing positions:<br>• Up to 60 different indexing positions can be stored in the table.<br>• The 1st entry in the table corresponds to indexing position 1; the nth entry thus to indexing position n.<br>• The indexing positions should 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.<br>• If the indexing axis is defined as a rotary axis (MD: IS_ROT_AX = “1”) with modulo 360° (MD: ROT_IS_MODULO = “1”) then the position values are limited to a range of 0° $\leqq$ pos. < 360°.<br><br>The number of indexing positions used in the table is defined by the MD: INDEX_AX_LENGTH_POS_TAB_1.<br>Entering the value 1 in the axial machine data INDEX_AX_ASSIGN_POS_TAB assigns indexing position table 1 to the current axis. | | | | |
| Application example(s) | Tool magazines (tool turrets, chain magazines) | | | | |
| Special cases, errors, ...... | Alarm 17020 “Illegal array index” if over 60 positions are entered in the table. | | | | |
| Related to .... | MD: INDEX_AX_ASSIGN_POS_TAB (axis is an indexing axis)<br>MD: INDEX_AX_LENGTH_POS_TAB_1 (number of indexing positions used in Table 1)<br>MD: IS_ROT_AX (rotary axis)<br>MD: ROT_IS_MODULO (modulo conversion for rotary axis) | | | | |

| 10930<br>MD number | INDEX_AX_POS_TAB_2 [n]<br>Indexing position table 2 [n] | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: *** | Maximum input limit: *** | |
| Modification effective after power ON or RESET with SW 4.3 and higher | Protection level: 2 / 7 | Unit: mm or degrees | |
| Data type: DOUBLE | Applies from SW: 1.1 | | |
| Meaning: | The indexing position table is used to assign the axis positions in the valid unit of measurement (mm, inches or degrees) to the indexing positions [n] on the indexing axis.<br>[n] = Index for the entry of the indexing positions in the indexing position table. Range: 0 ≧ n ≦ 59, where 0 is the 1st indexing position and 59 the 60th indexing position.<br>Note: Programming with the absolute indexing position (e.g. CAC) starts with indexing position 1. This corresponds to the indexing position with index n = 0 entered in the table of indexing positions.<br>The following should be noted when entering the indexing positions:<br>• Up to 60 different indexing positions can be stored in the table.<br>• The 1st entry in the table corresponds to indexing position 1; the nth entry thus to indexing position n.<br>• The indexing positions should 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.<br>• If the indexing axis is defined as a rotary axis (MD: IS_ROT_AX = “1”) with modulo 360° (MD: ROT_IS_MODULO = “1”) then the position values are limited to a range of 0° ≦ pos. < 360°.<br><br>The number of indexing positions used in the table is defined by the MD: INDEX_AX_LENGTH_POS_TAB_2.<br>Entering the value 1 in the axial machine data INDEX_AX_ASSIGN_POS_TAB assigns indexing position table 1 to the current axis. | | |
| Application example(s) | Tool magazines (tool turrets, chain magazines) | | |
| Special cases, errors, ...... | Alarm 17020 “Illegal array index” if over 60 positions are entered in the table. | | |
| Related to .... | MD: INDEX_AX_ASSIGN_POS_TAB (axis is an indexing axis)<br>MD: INDEX_AX_LENGTH_POS_TAB_2 (number of indexing positions used in Table 2)<br>MD: IS_ROT_AX (rotary axis)<br>MD: ROT_IS_MODULO (modulo conversion for rotary axis) | | |

| 10940<br>MD number | INDEX_AX_MODE<br>Options for indexing position | | |
|---|---|---|---|
| Default setting: 0 | Minimum input limit: 0 | Maximum input limit: 1 | |
| Changes effective after POWER ON | Protection level: 2 / 7 | Unit: – | |
| Data type: DWORD | Applies from SW: 7.2 | | |
| Meaning: | The machine data controls the display of indexing positions in system variable AA_ACT_INDEX_AX_POS_NO and OPI variable aaActIndexAxPosNo.<br>Bit 0 = 0: Indexing position display changes when indexing position is reached/crossed (indexing area is between the indexing positions, compatible behavior).<br>Bit 0 = 1: Indexing position display changes when half the indexing axis position is crossed (indexing area is quasi-symmetrical around the indexing position). | | |
| Related to .... | System variable $AA_ACT_INDEX_AX_POS_NO | | |

## 4.2 Axis-specific machine data

| | | | | | |
|---|---|---|---|---|---|
| **30500**<br>MD number | **INDEX_AX_ASSIGN_POS_TAB**<br>Axis is indexing axis | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 3 | |
| Modification effective after power ON or RESET with SW 4.3 and higher | | | Protection level: 2 / 7 | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 1.1 | |
| Meaning: | The axis is declared as an indexing axis by assignment of indexing position table 1 or 2.<br>0: The axis is not declared as an indexing axis.<br>1: The axis is an indexing axis. The associated indexing positions are stored in Table 1 (MD: INDEX_AX_POS_TAB_1).<br>2: The axis is an indexing axis. The associated indexing positions are stored in Table 2 (MD: INDEX_AX_POS_TAB_2).<br>3: Equidistant indexing with SW 4.3 and higher (840D) and SW 2.3 and higher (810D)<br>>3: Alarm 17090 "Value violates upper limit" | | | | |
| Application example(s) | Tool magazines (tool turrets, chain magazines) | | | | |
| Special cases, errors, ...... | 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). Alarm 4000 is otherwise generated at boot.<br>Alarm 17500 "Axis is not an indexing axis"<br>Alarm 17090 "Value violates upper limit" | | | | |
| Related to .... | MD: INDEX_AX_POS_TAB1 (indexing position table 1)<br>MD: INDEX_AX_LENGTH_POS_TAB_1 (no. of indexing positions used in table 1)<br>MD: INDEX_AX_POS_TAB2 (indexing position table 2)<br>MD: INDEX_AX_LENGTH_POS_TAB_2 (no. of indexing positions used in table 2)<br>For **equidistant indexes with** a value of 3:<br>MD: INDEX_AX_NUMERATOR Numerator<br>MD: INDEX_AX_DENOMINATOR Denominator<br>MD: INDEX_AX_OFFSET First indexing position<br>MD: HIRTH_IS_ACTIVE Hirth tooth system | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **30501**<br>MD number | **INDEX_AX_NUMERATOR**<br>Numerator for indexing axes with equidistant positions | | | | |
| Default setting: 0 | | Minimum input limit: >0 | | Maximum input limit: *** | |
| Changes effective after RESET | | | Protection level: 2/7 | | Unit: mm/inches/degrees |
| Data type: DOUBLE | | | | Applies from SW: 4.3 | |
| Meaning: | Defines the value of the numerator for calculating the distances between two indexing positions when the positions are equidistant. Modulo axes ignore this value and use $MA_MODULO_RANGE instead. | | | | |
| MD irrelevant for ...... | Non-equidistant indexes in accordance with tables | | | | |
| Application example(s) | See 2.4 | | | | |
| Related to .... | MD 30502: INDEX_AX_DENOMINATOR, MD 30503: INDEX_AX_OFFSET; MD 30500: INDEX_AX_ASSIGN_POS_TAB | | | | |

| | | | | |
|---|---|---|---|---|
| **30502**<br>MD number | **INDEX_AX_DENOMINATOR**<br>Denominator for indexing axes with equidistant positions | | | |
| Default setting: 1 | | Minimum input limit: 1 | | Maximum input limit: *** |
| Changes effective after RESET | | Protection level: 2/7 | | Unit: – |
| Data type: DWORD | | | Applies from SW: 4.3 | |
| Meaning: | Defines the value of the denominator for calculating the distances between two indexing positions when the positions are equidistant. For modulo axes it therefore specifies the number of indexing positions. | | | |
| MD irrelevant for ...... | Non-equidistant indexes in accordance with tables | | | |
| Application example(s) | See 2.4 | | | |
| Related to .... | MD 30501: INDEX_AX_NUMERATOR, MD 30503: INDEX_AX_OFFSET; MD 30500: INDEX_AX_ASSIGN_POS_TAB | | | |

| | | | | |
|---|---|---|---|---|
| **30503**<br>MD number | **INDEX_AX_OFFSET**<br>First indexing position for indexing axes with equidistant positions | | | |
| Default setting: 0.0 | | Minimum input limit: *** | | Maximum input limit: *** |
| Changes effective after RESET | | Protection level: 2/7 | | Unit: mm/inches/degrees |
| Data type: DOUBLE | | | Applies from SW: 4.3 | |
| Meaning: | Defines the position of the first indexing position from zero for an indexing axis with equidistant positions. | | | |
| MD irrelevant for ...... | Non-equidistant indexes in accordance with tables | | | |
| Application example(s) | See 2.4 | | | |
| Related to .... | MD 30501, 30502, 30500 | | | |

| | | | | |
|---|---|---|---|---|
| **30505**<br>MD number | **HIRTH_IS_ACTIVE**<br>Hirth tooth system is active | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 |
| Changes effective after RESET | | Protection level: 2/7 | | Unit: – |
| Data type: BOOLEAN | | | Applies from SW: 4.3 | |
| Meaning: | Hirth tooth system is active when a value of 1 is set. | | | |
| MD irrelevant for ...... | Non-equidistant indexes in accordance with tables | | | |
| Application example(s) | See 2.4 | | | |
| Related to .... | MD 30500, 30501, 30502, 30503 | | | |

■

# Signal Descriptions 5

## 5.1 Axisspecific signals

| | | | |
|---|---|---|---|
| **DB31, ...**<br>**DBX76.6**<br>Data Block | **Indexing axis in position**<br>Signal(s) from axis/spindle (NCK —> PLC) | | |
| Edge evaluation: no | | Signal(s) updated: Cyclic | Signal(s) valid from SW: 1.1 |
| Signal state 1 or signal transition 0 ——> 1 | The signal is influenced according to the “Exact stop fine”.<br>When “Exact stop fine” is achieved, the signal is set. When exiting “Exact stop fine”, the signal is reset again.<br>• The indexing axis is at an indexing position. The indexing axis has been positioned by coded position instructions.<br>SW 4.3 and higher (840D), SW 2.3 and higher (810D)<br>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. | | |
| Signal state 0 or signal transition 1 ——> 0 | • The axis is not defined as an indexing axis.<br>• The indexing axis is traversing (IS “Travel command +/–” (DB31, ... DBX64.7/64.6) is active)<br>• The indexing axis is located at a position which is not an indexing position.<br>Examples: In JOG mode after abortion of travel movement, e.g. with RESET<br>In Automatic mode: Indexing axis has, for example, approached a selected position controlled by an AC or DC instruction.<br>• The indexing axis has not been positioned with instructions for coded positions (CAC, CACP, CACN, CDC, CIC) in automatic mode.<br>• The “Servo enable” signal for the indexing axis has been canceled (IS “Servo enable” DB31, ... DBX2.1). | | |
| Signal irrelevant for ...... | Axes that are not defined as indexing axes<br>(MD 30500: INDEX_AX_ASSIGN_POS_TAB = “0”) | | |
| Application example(s) | Tool magazine: the activation of a gripper for removing a tool from a magazine is triggered when the indexing axis is in position (“indexing axis in position” = 1).<br>This must be programmed in the PLC user program. | | |
| Special cases, errors, ...... | Notes:<br>The axis positions entered in the indexing position table for the individual divisions can be changed through zero offsets (including DRF). The “indexing axis in position” interface signal 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. For exceptions, see 2.4 Hirth tooth system. | | |
| Related to .... | MD 30500: INDEX_AX_ASSIGN_POS_TAB (axis is an indexing axis) | | |

■

## Notes

# Example 6

For an example, please see Sections 2.4, 2.5

■

# Data Fields, Lists 7

## 7.1 Interface signals

| DB number | Bit, byte | Name | Refer-ence |
|---|---|---|---|
| Axis/spindle-specific | | | |
| 31–61 | 76.6 | Indexing axis in position | |
| 31–61 | 60.4, 60.5 | Referenced/synchronized 1, referenced/synchronized 2 | R1 |

## 7.2 Machine data

| Number | Names | Name | Refer-ence |
|---|---|---|---|
| General ($MN_ ... ) | | | |
| 10260 | CONVERT_SCALING_SYSTEM | Basic system switchover active | G2 |
| 10270 | POS_TAB_SCALING_SYSTEM | System of measurement of position tables | |
| 10900 | INDEX_AX_LENGTH_POS_TAB_1 | Number of indexing positions used in Table 1 | |
| 10920 | INDEX_AX_LENGTH_POS_TAB_2 | Number of indexing positions used in Table 2 | |
| 10910 | INDEX_AX_POS_TAB_1[n] | Indexing position table 1 | |
| 10930 | INDEX_AX_POS_TAB_2 [n] | Indexing position table 2 | |
| 10940 | INDEX_AX_MODE | Options for indexing positions | |
| Axis/channelspecific ($MA_ ... ) | | | |
| 30300 | IS_ROT_AX | Rotary axis | R2 |
| 30310 | ROT_IS_MODULO | Modulo conversion for rotary axis | R2 |
| 30320 | DISPLAY_IS_MODULO | Position display “Modulo 360°” | R2 |

| Number | Names | Name | Reference |
|---|---|---|---|
| 30500 | INDEX_AX_ASSIGN_POS_TAB | Axis is indexing axis | |
| 30501 | INDEX_AX_NUMERATOR | Counter 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 | |

## 7.3 Setting data

| Number | Names | Name | Reference |
|---|---|---|---|
| General ($SN_ ...) | | | |
| 41050 | JOG_CONT_MODE_LEVELTRIGGRD | JOG continuous mode | H1 |

## 7.4 System variables

The following system variables exist in SW 4.3 and higher:

| Names | Name, meaning | Reference |
|---|---|---|
| $AA_ACT_INDEX_AX_POS_NO[axis] | Number of last indexing position reached or overtraveled | PGA 1 |
| $AA_PROG_INDEX_AX_POS_NO[axis] | Number of programmed indexing position | PGA 1 |

## 7.5 Alarms

For detailed descriptions of the alarms, please refer to
**References:** /DA/, “Diagnostics Guide”
and the online help of MMC 101/102/103 systems.

■

# SINUMERIK 840D/840Di/810D
# Description of Functions
# Extended Functions (FB2)

# Tool Change (W3)



■

# Notes

# 1 Brief Description

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:

- Movement of the tool carrier from the machining position to the tool change position
- Tool change
- Movement of the tool carrier from the tool change position to the new machining position

**Control**

The tool change can be actuated using a

- T function or an
- M command (preferably M06)

There are two options for tool change:

Immediate change with T number or preparation with T number:

1. Immediate change
   - The T function loads the new tool immediately.
   - Typical application: Turning machines with tool turret
2. Preparation
   - The new tool is prepared for the change on execution of the T function.
   - The M function is used to remove the old tool from the spindle and load the new tool.
   - The M command for tool change can be defined in a machine data.
   - Typical application: Milling machines with a tool magazine, in order to bring the new tool into the tool change position without interrupting the machining process.

**Tool change point**

The selection of the tool change point has a significant effect on the cut-to-cut time. The tool change point is chosen according to the machine tool concept and, in certain cases, according to the current machining task.

Fixed positions on a machine axis stored in machine data can be approached by means of the **Fixed-point approach** function G75. This can be used to define and control one or several tool change points.

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.

■

# Notes

# Detailed Description 2

## 2.1 Overview of tool change function

CNC-controlled machine tools are equipped with tool magazines and automatic tool change facility for the complete machining of workpieces.

**Tool changing equipment**

Tool magazines and tool changing equipment are selected according to the machine type.

Turning machines: Tool turret (disc, flat or inclined tool turret); no special equipment required: Tool is changed through rotation of revolver.

Milling machines: Magazines (chain, plate, disc, cassette) with gripper/double gripper for changing.

Since the tool change interrupts machining, the idle times must be minimized.

**Tool change times**

Tool change times are largely determined by the construction of the machine tool.

Typical tool change times are

| | |
|---|---|
| 0.1 to 0.2 secs | for rotation of a tool turret |
| 0.3 to 2 secs | for a tool change with a gripper for an available tool. |

**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 on the interruption point (return to cut) with the new tool when the spindle is rotating.

Typical cut-to-cut times are:

| | |
|---|---|
| 0.3 to 1 secs | for a turning machine with tool turret |
| 0.5 to 5 secs | for a milling machine with a tool changer. |

**Requirements**

A tool change operation must fulfill the following requirements:

– Short idle times
– Fast search, preparation and return of the tool during machining
– Simple programming of the tool change cycle

– Automatic operation of the required axis and gripper movements
– Easy fault recovery

## 2.2 Sequence

**Tool change sequence**

The procedure for changing tools comprises three steps:

– Movement of the tool carrier from the machining position to the tool change position
– Tool change
– Movement of the tool carrier from the tool change position to the new machining position

The tool change position depends on the machine concept and is described in more detail in Section 2.4.

**Control of spindle**

The method by which the spindle is controlled during a tool change also depends on the machine design. The various options include systems where

– the spindle continues to rotate
– the spindle is brought to a halt, or
– the spindle is positioned

## 2.3 Control

**Control**

The tool change can be actuated using a

– T function or an
– M command (preferably M06).

The selection is made in MD: TOOL_CHANGE_MODE, as follows:

TOOL_CHANGE_MODE = 0

– The T function loads the new tool immediately.
– Typical application: Turning machines with tool turret

TOOL_CHANGE_MODE = 1

– The new tool is prepared for the change on execution of the T function.
– The M function is used to remove the old tool from the spindle and load the new tool.
– The M command for the tool change is defined in the MD: TOOL_CHANGE_M_CODE. The default setting is 6 for compatibility with DIN 66025.

– Typical application: Milling machines with a tool magazine, in order to bring the new tool into the tool change position without interrupting the machining process.

Note: If the tool offset number is supplied from the PLC or an MMC tool manager, a STOPRE block search stop must be inserted at a suitable point. STOPRE must be avoided, however, when tool radius compensation (G41/G42) or SPLINE interpolation are active, since several blocks are required here in advance for the path calculation.

Further information on M functions which apply to the M06 tool change, such as

– Extended address
– Output time to PLC
– Auxiliary function groups
– Block search response
– Response to overstore

can be found in:
**References**: /FB/, S5, "Synchronized Actions"

## 2.4 Tool change point

**Tool change point**

The selection of the tool change point has a significant effect on the cut-to-cut time. The tool change point is chosen according to the machine tool concept and, in certain cases, according to the current machining task.

The **fixed point approach** function (G75) can be used to approach fixed positions on a machine axis:

N20 G75 FP=2 X1=0 Y1=0 Z1=0 LF

**Fixed points**

Two fixed positions are stored for each machine axis in MD: FIX_POINT_POS[N]. They are addressed with FP=1 or FP=2. If no value is defined, then FP=1.

Each machine axis which is required to travel to one of these points has to be specified with its machine name and a dummy position (which is not evaluated).

The positions stored in the MD are approached with rapid traverse G0.

In a block with G75, the spindle can be positioned using SPOS and SPOSA.

■

## Notes

# Supplementary Conditions 3

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.

■

# Data Descriptions (MD, SD) 4

## 4.1 Machine data

The machine data required for the tool change are documented in the following publications:

| MD number | Identifier | Description of Functions |
|---|---|---|
| 22500 | TOOL_CHANGE_MODE | W1 |
| 22600 | TOOL_CHANGE_M_CODE | W1 |
| 30600 | FIX_POINT_POS[n] | K1 |
| 22200 | AUXFU_M_SYNC_TYPE | H2 |
| 22220 | AUXFU_T_SYNC_TYPE | H2 |

■

# Signal Descriptions 5

No separate signals exist for this Description of Functions.

■

# Notes

# 6 Example

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

```
N 970    G0  X=  Y=  Z=  LF; Retract from the contour
N 980    T1  LF                    ; Tool selection
N 990    W_WECHSEL  LF             ; Subroutine call without parameters
N 1000   G90  G0  X=  Y=  Z=  M3  S1000  LF; Machining resumed
```

Subroutine for tool change

```
PROC  W_WECHSEL  LF
N 10     SPOSA=  S0  LF                ; Spindle positioning
N 20     G75  FP=2  X1=0  Y1=0  Z1=0;    Approach tool
                                         change point (see Section 2.4)
N 30     M06  LF                       ; Change tools
N 40     M17  LF
```

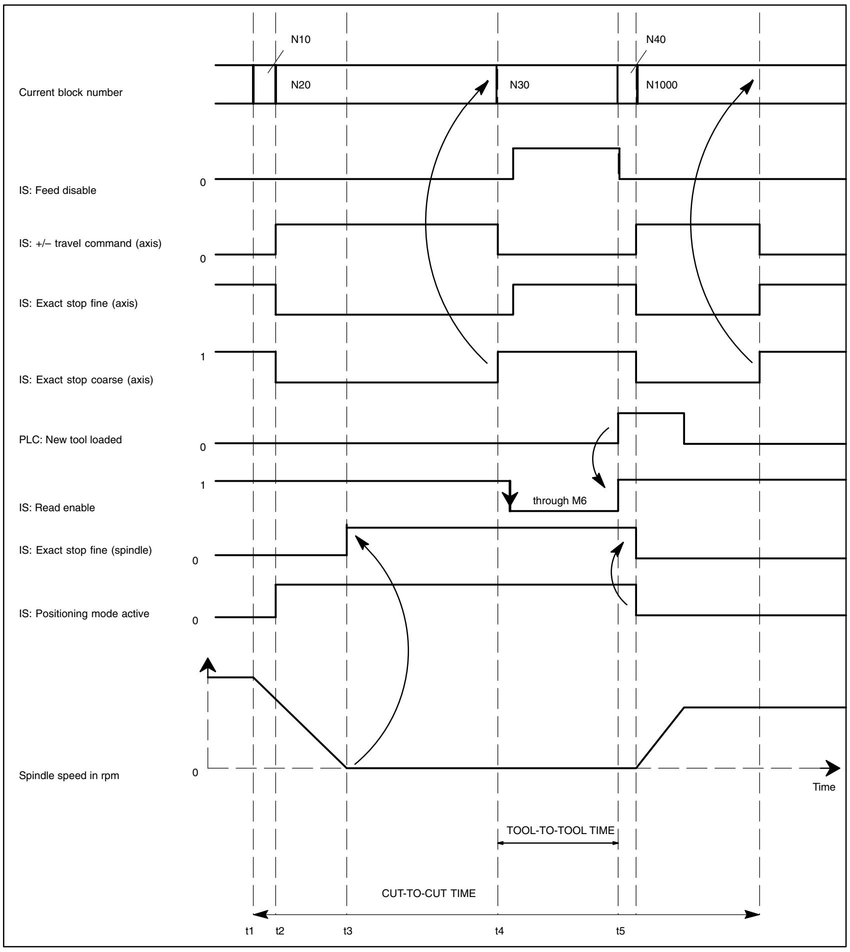



Fig. 6-1 Chronological sequence of tool change

t1 Axes stationary
Spindle rotating
Start of tool change cycle in N 10

t2 Traverse axis with G75 in N 20 to tool change point

t3 Spindle reaches programmed position from block N 10

t4 Axes reach fine stop coarse from N 20; N 30 starts here:
N 30:

M06 removes the previous tool from the spindle and loads and secures the new tool.

t5 Tool changer swivels back to original position.

Following this, in N 1000 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

■

# 7 Data Fields, Lists

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| Channel-specific | | | |
| 21–28 | 194.6 | M function M06 | |

## 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| General ($MN_ ...) | | | |
| 18082 | MM_NUM_TOOL | Number of tools | S7 |
| Channel-specific ($MC_ ... ) | | | |
| 22200 | AUXFU_M_SYNC_TYPE | Output timing for M functions | H2 |
| 22220 | AUXFU_T_SYNC_TYPE | Output timing of T functions | H2 |
| 22550 | TOOL_CHANGE_MODE | New tool offset for M function | |
| 22560 | TOOL_CHANGE_M_CODE | M function for tool change | |
| Axis-specific ($MA_ ...) | | | |
| 30600 | FIX_POINT_POS[n]. | Fixed point positions of the machine axes with G75 | |

## 7.3 Interrupts

For detailed descriptions of the alarms, please refer to
**References:** /DA/, “Diagnostics Guide”
and the online help of MMC 101/102/103 systems.

■

# Notes

# SINUMERIK 840D/840Di/810D
# Description of Functions
# Extended Functions (FB2)

# Tool Compensation and Monitoring in Grinding (W4)

■

# Brief Description 1

**Contents**

This Description of Functions deals with the following subjects:

- Tool offset for grinding operations
- Online tool offsets (continuous dressing)
- Grinding-specific tool monitoring
- Constant grinding wheel peripheral speed (GWPS)

---

**Note**

This Description is based on information in
**References:** /FB/, W1, “Tool Offset”

For information about programming, mode of operation and handling, please refer to
**References**: /PG/, Programming Guide Fundamentals

---

■

## Notes

# Detailed Description 2

## 2.1 Tool offset for grinding operations

### 2.1.1 Structure of tool data

**Grinding tools**

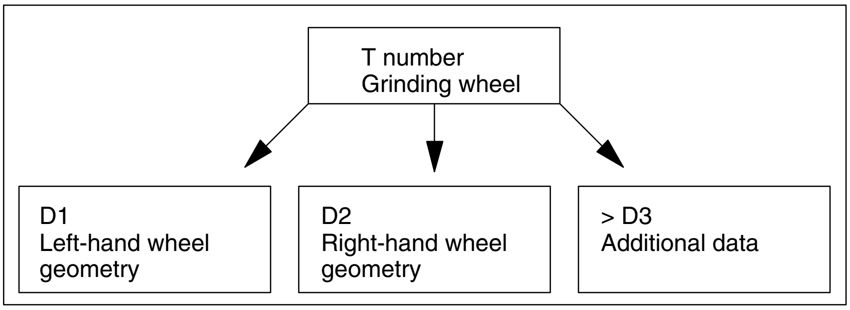
Grinding tools (see Subsection 2.1.4 and Chapter 3) are tools of type 400 to 499.

**Tool offset for grinding tools**

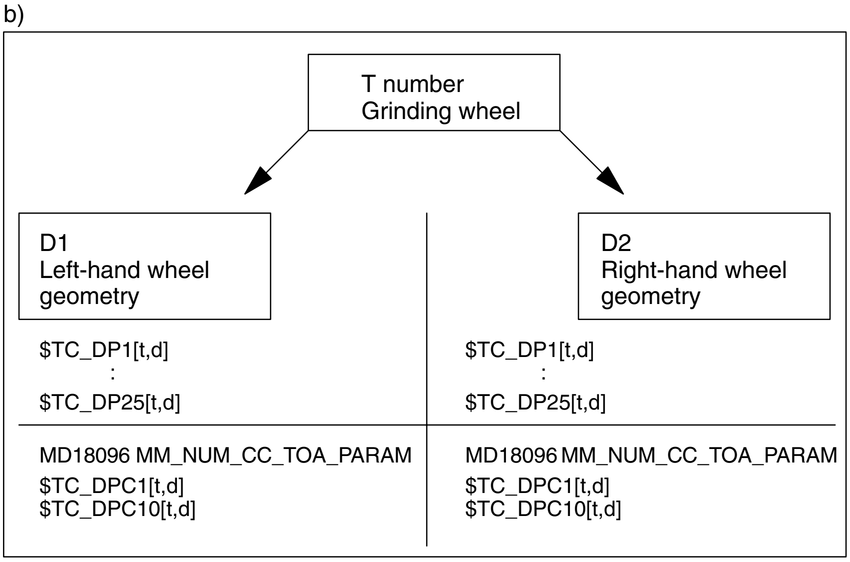
Apart from edge-specific data, data that are specific to the tool and dressing process are generally also programmed for grinding tools.
The data specific to a grinding wheel for the left-hand and right-hand wheel geometry can be stored in D1 or D2 under a T number.
If data are required for the dressing geometry, they can be stored, for example, starting at D3 of a T number or in additional edge-specific data (MD18096 MM_NUM_CC_TOA_PARAM).

**Examples:**

a)




b)



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

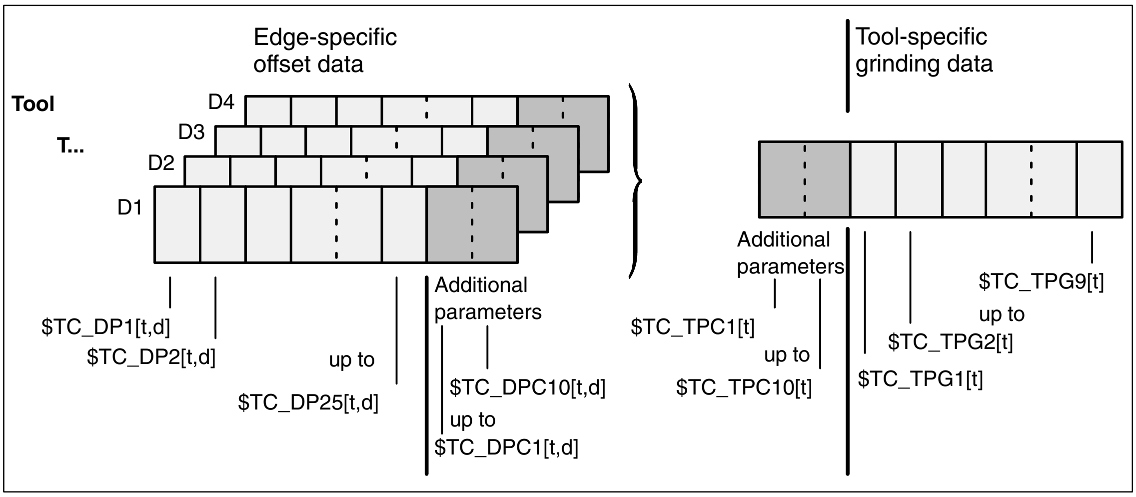



Fig. 2-1 Structure of tool offset data for grinding tools

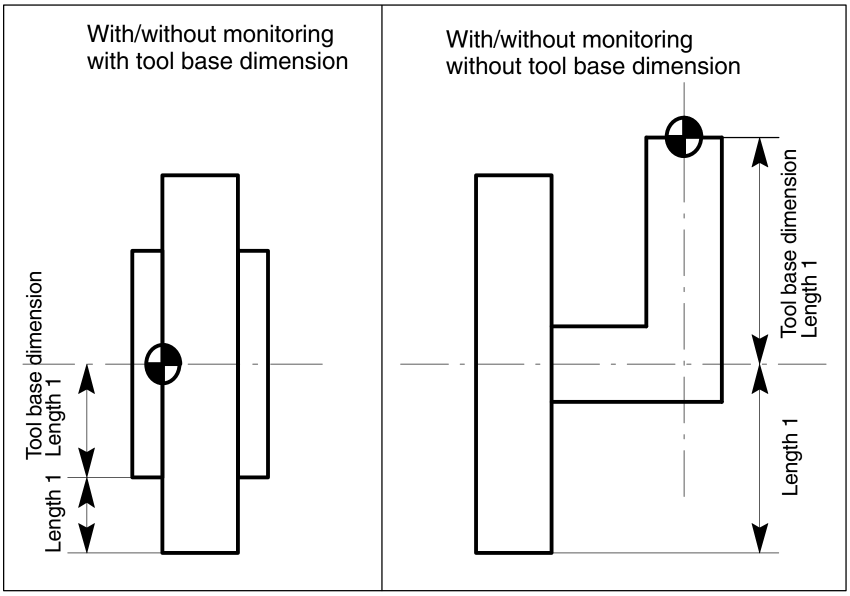
### 2.1.2 Edge-specific offset data

**Tool parameters**

The tool parameters for grinding tools have the same meaning as those for turning and milling tools.

| Tool parameter | Meaning | Remarks | Reserved for expansions |
|---|---|---|---|
| 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 | | |
| Wear tool radius compensation | | | |
| 15 | Radius 1 | | |
| 16 | | | Reserved |
| 17 | | | Reserved |
| 18 | | | Reserved |
| 19 | | | Reserved |
| 20 | | | Reserved |
| Base dimension/adapter dimension tool length compensation | | | |
| 21 | Basic length 1 | | |
| 22 | Basic length 2 | | |
| 23 | Basic length 3 | | |
| Technology | | | |
| 24 | Undercut angle | only for turning tools | |
| 25 | | | Reserved |

Reserved ... means that this tool parameter of the 840D/810D is not used (reserved for expansions).




---

**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, the additional parameters $TC_DPC1 to 10 can be set up using general MD18096: MM_NUM_CC_TOA_PARAM, regardless of the tool type.

---

**Caution**

Changes to the MD take effect after POWER ON and will lead to initialization of the memory (back data up beforehand if necessary!).

An automatic changeover between grinding wheel offset left and right does **not** take place during contour grinding. This changeover must be programmed.

---

**Tool types for grinding tools**

The structure of tool types for grinding tools is as follows:

Tool type 4 X Y

Y:
0: Without monit. with base dimension for GWPS
1: With monit. with base dimension for GWPS
2: Without monit. without base dim. for GWPS
3: With monit. without base dim. for GWPS

X:
0: Surface grinding wheel
1: Facing wheel
9: Dresser

Fig. 2-2 Structure of tool type for grinding tools, see Fig. 2–1

---

**Note**

Channel-specific MD20350: TOOL_GRIND_AUTO_TMON can be set to define whether the monitoring function must already be active when tools **with monitoring** (i.e. uneven tool types) are selected.

---

**Examples:**
This structure can be used to create the following tool types:

- Type 400: Surface grinding wheel
- Type 401: Surface grinding wheel with monitoring and tool base dimension for GWPS
- Type 403: Surface grinding wheel with monitoring/without tool base dimension for GWPS
- Type 410: Facing wheel
- Type 411: Facing wheel with monitoring with base dimension for GWPS
- Type 413: Facing wheel with monitoring without base dimension for GWPS
- Type 490: Dresser

## 2.1.3 Tool-specific grinding data

Tool-specific grinding data are available once for every T number (type 400-499). They are automatically set up with every new grinding tool (type 400-499).

---

**Note**

Tool-specific grinding data have the same characteristics as a tool edge. This may need to be taken into account when the number of cutting edges is specified in MD18100: MM_NUM_CUTTING_EDGES_IN_TOA.

---

When all the cutting edges of a tool are deleted, the existing tool-specific grinding data are deleted at the same time.

**Tool-specific grinding data**

The parameters are assigned as follows:

| Parameters | 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 |
| Additional parameters (user-specific cutting edge data) | | |
| $TC_TPC1 | | Real |
| up to | | |
| $TC_TPC10 | | Real |

**Definition of additional parameters $TC_DPC1...10**

For user-specific cutting edge data, the additional parameters $TC_DPC1 to 10 can be set up using general MD 18096: MM_NUM_CC_TDA_PARAM, regardless of the tool type concerned.

**Caution**

Changes to the MD take effect after POWER ON and will lead to initialization of the memory (back data up beforehand if necessary!).

**Spindle number $TC_TPG1**

Number of programmed spindle (e.g. grinding wheel peripheral speed) and spindle to be monitored (e.g. wheel radius and width)

**Chaining rule $TC_TPG2**

This parameter is set to define which tool parameters of tool edge 2 (D2) and tool edge 1 (D1) must 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 |

| Tool parameter | Meaning | Bit in $TC_TPG2 | Hex | Dec |
|---|---|---|---|---|
| $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 |
| 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 |
| 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:

a) Binary
$TC_TPG2[1]= 'B111 0000 0011 1000 0001 1101'
(Bit 22 ... Bit 0)
b) Hexadecimal
$TC_TPG2[1]= 'H70381D'
c) 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.

---

**Min. wheel radius and width**
**$TC_TPG3**
**$TC_TPG4**

The limit values for the grinding wheel radius and width must be entered in this parameter. 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. The user can use these signals to define his error strategy.

---

**Current width**
**$TC_TPG5**

The width of the grinding wheel measured, for example, after the dressing operation, is entered here.

**Max. 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 this parameter.

**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.

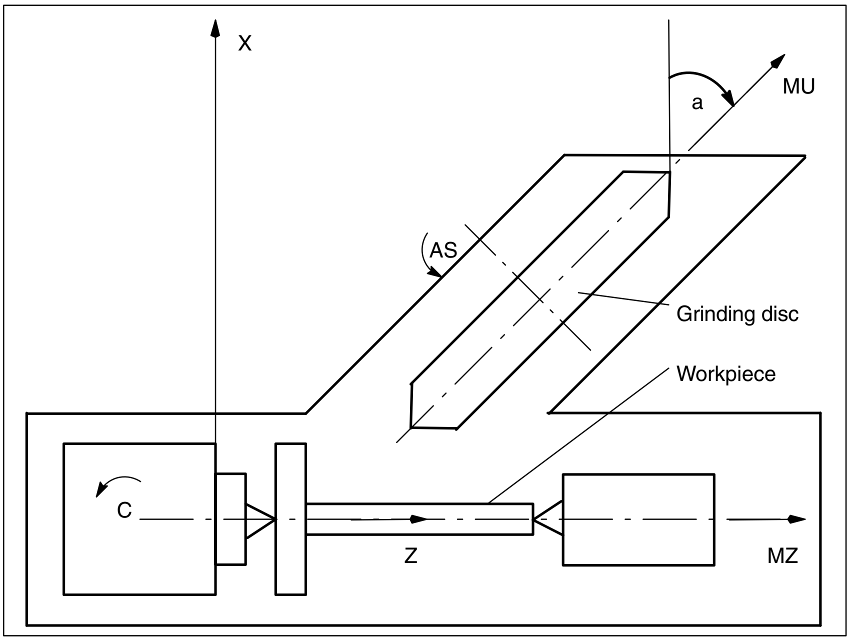



Fig. 2-3 Machine with inclined infeed axis

**Note**

The tool lengths are not automatically compensated when the angle is altered.

The angle must be within the $-90 \leq$ $TC_TPG8 $<$ +90 range.

In the case of machines with inclined axes, the same angle must be set for the inclined axis and the inclined grinding wheel. With SW versions NCU 5.1.11/P5 and 4.4.35/P4 and earlier with GWPS for grinding wheel cutting rate and inclined axis, the input value in tool parameter $TC_TPG8 must be set in RAD rather than DEGREE. $TC_TPG8 in RAD = PI/180* angle.

**Parameter number for radius calculation $TC_TPG9**

This parameter specifies which offset values are to be used for the GWPS calculation and tool monitoring for 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 parts program**

Parameters can be read and written from the parts 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 apply to both the left-hand and the right-hand cutting edge of the grinding wheel.
- The tool-specific grinding data take effect when the following are programmed:
  GWPSON (selection of constant wheel peripheral speed), TMON (selection of tool monitoring function). To make a changed data effective, GWPSON or TMON must be programmed 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).

---

### 2.1.4 Examples of grinding tools

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 |

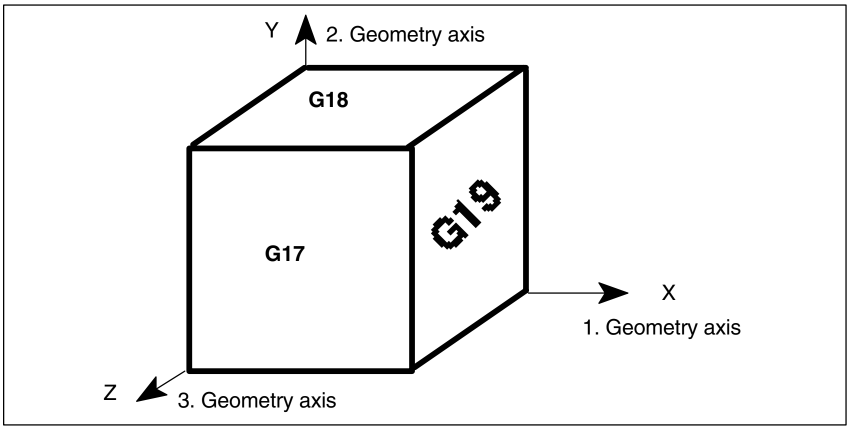


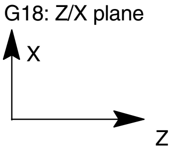

Fig. 2-4 Planes and axis assignment

### 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




Effect

| | |
|---|---|
| G17 | Length 1 in Y<br>Length 2 in X<br>Radius in X/Y |
| G18 | Length 1 in X<br>Length 2 in Z<br>Radius in Z/X |
| G19 | Length 1 in Z<br>Length 2 in Y<br>Radius in Y/Z |

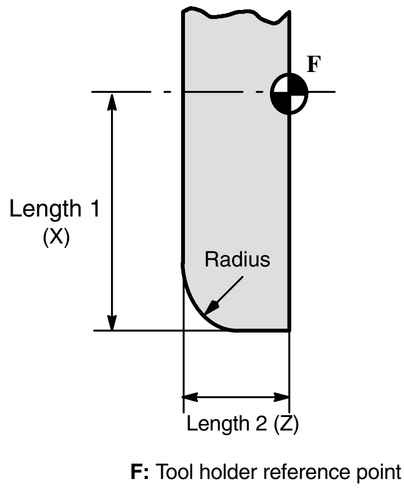



Fig. 2-5 Offset values required by a surface grinding wheel

**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.

Effect

| | |
|---|---|
| G17 | Length 1 in Y<br>Length 2 in X<br>Radius in X/Y |
| G18 | Length 1 in X<br>Length 2 in Z<br>Radius in Z/X |
| G19 | Length 1 in Z<br>Length 2 in Y<br>Radius in Y/Z |

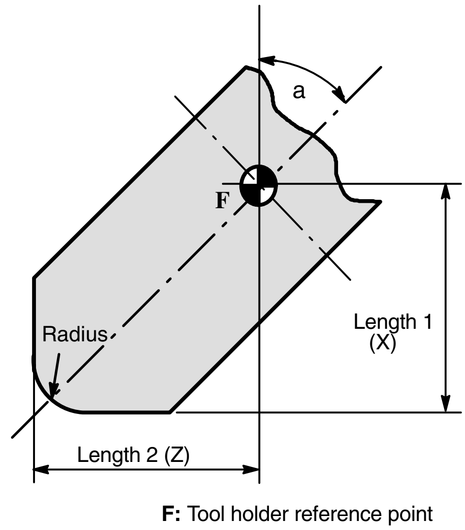



Fig. 2-6 Offset values required for inclined wheel with implicit monitoring selection

**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.

Effect

| | |
|---|---|
| G17 | Length 1 in Y<br>Length 2 in X<br>Radius in X/Y |
| G18 | Length 1 in X<br>Length 2 in Z<br>Radius in Z/X |
| G19 | Length 1 in Z<br>Length 2 in Y<br>Radius in Y/Z |

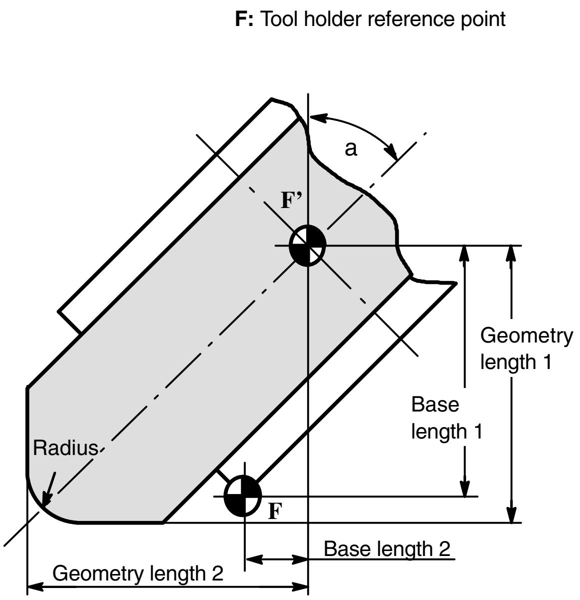



Fig. 2-7 Required offset values shown by example of inclined grinding wheel with implicit monitoring selection and with base selection for GWPS calculation

### 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.

| Effect | |
|---|---|
| G17 | Length 1 in Y<br>Length 2 in X<br>Radius in X/Y |
| G18 | Length 1 in X<br>Length 2 in Z<br>Radius in Z/X |
| G19 | Length 1 in Z<br>Length 2 in Y<br>Radius in Y/Z |

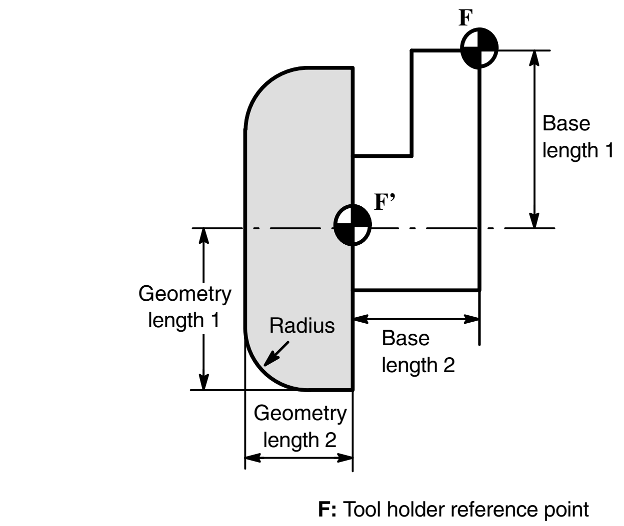



Fig. 2-8 Required offset values of a surface grinding wheel without base dimension for GWPS

### 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.

| Effect | |
|---|---|
| G17 | Length 1 in Y<br>Length 2 in X<br>Radius in X/Y |
| G18 | Length 1 in X<br>Length 2 in Z<br>Radius in Z/X |
| G19 | Length 1 in Z<br>Length 2 in Y<br>Radius in Y/Z |

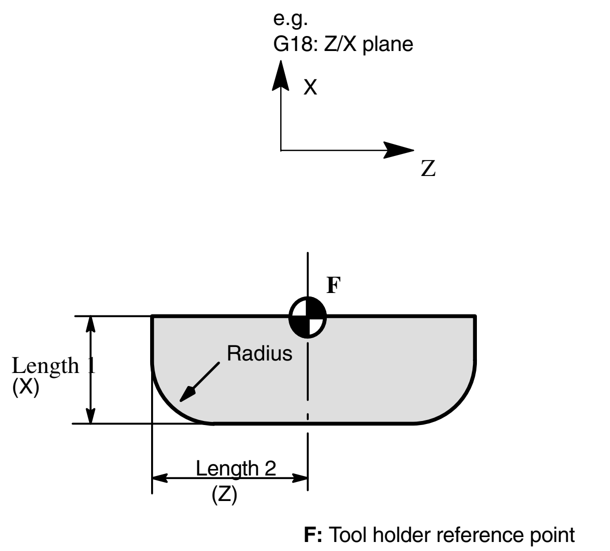



Fig. 2-9 Required offset values of a facing wheel with monitoring parameters

## 2.2 Online tool offset

**Application**

A grinding operation involves both machining of a workpiece and dressing of the grinding wheel. These processes can be operated 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 (see Chapter 3).

**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 (see Chapter 3).

---

**Note**

The online tool offset may only be used for grinding tools.

---

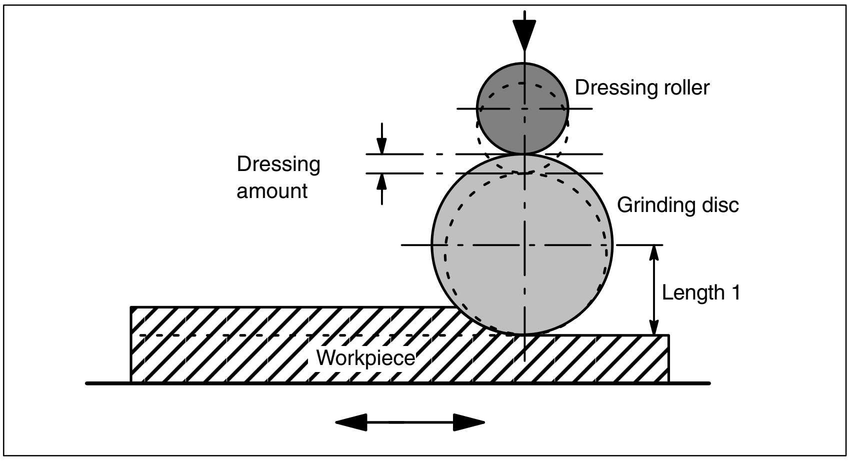



Fig. 2-10 Dressing with a dressing roller in parallel to machining

**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 (see Section 2.3) 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 (<SW4).

Online offsets are also applied to the constant grinding wheel peripheral speed (GWPS), i.e. the spindle speed is corrected by the corresponding value.

---

**Commands**

The following commands are provided for online tool offsets:

| Command | Meaning |
|---|---|
| **FCTDEF**<br>(<polynomial_no>, <lower_limit_value>, <upper_limit_value>, <coefficient 0>, <coefficient 1>, <coefficient 2>, <coefficient 3>) | Parameterize function (up to 3rd degree polynomial)<br>(Fine Tool Offset Definition) |
| **PUTFTOCF**<br>( <polynomial_no>, <reference_value>, <Length1_2_3>, <channel_no>, <spindle_no>) | Write online tool offset continuously<br>(Put Fine Tool Offset Compensation) |
| **PUTFTOC**<br>(<value>, <Length1_2_3>, <channel_no>, <spindle_no>) | Write online tool offset discretely<br>(Put Fine Tool Offset Compensation) |
| **FTOCON** | Activation of online tool offset<br>(Fine Tool Offset Compensation ON) |
| **FTOCOF** | Deactivation of online tool offset<br>(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:** /PA/, Programming Guide

---

## 2.2.1 Write online tool offset: Continuous

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:

**Parameterize function**

The function parameters are set in a separate block according to the following syntax:

**FCTDEF(<polynomial_no.>, <lower_limit_value>, <upper_limit_value>, <coefficient $a_0$>, <coefficient $a_1$>, <coefficient $a_2$>, <coefficient $a_3$>)**

| | |
|---|---|
| FCTDEF | Function Definition |
| Polynomial no.: | Number of function (e.g. 1, 2 or 3) |
| Lower/upper limit value: | Determines value range of function (limit values in input resolutions) |
| Coefficients $a_0$, $a_1$, $a_2$ | Coefficients of polynomial |

A polynomial of the 3rd degree is generally defined as follows:
$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3$

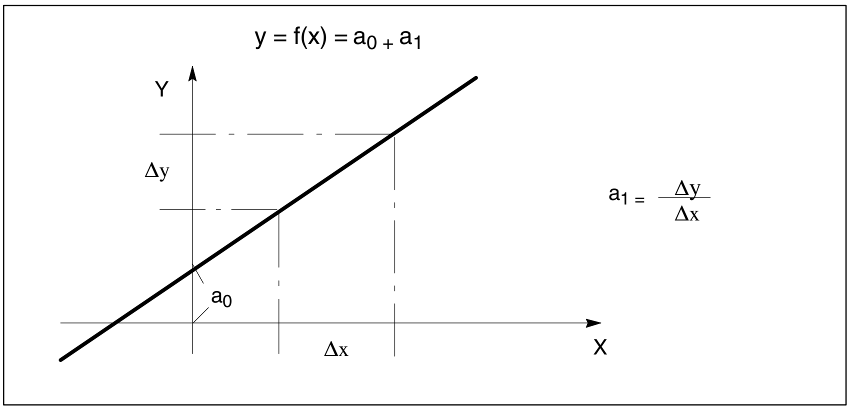



Fig. 2-11 Straight line equation

**Note**

FCTDEF must be programmed in a separate NC block.

**Example:**

Let us assume: Pitch a1 = +1
a2 = 0
a3 = 0

The value of the function must be y = 0 at the instant of definition and be derived from machine axis XA (e.g. dresser axis).

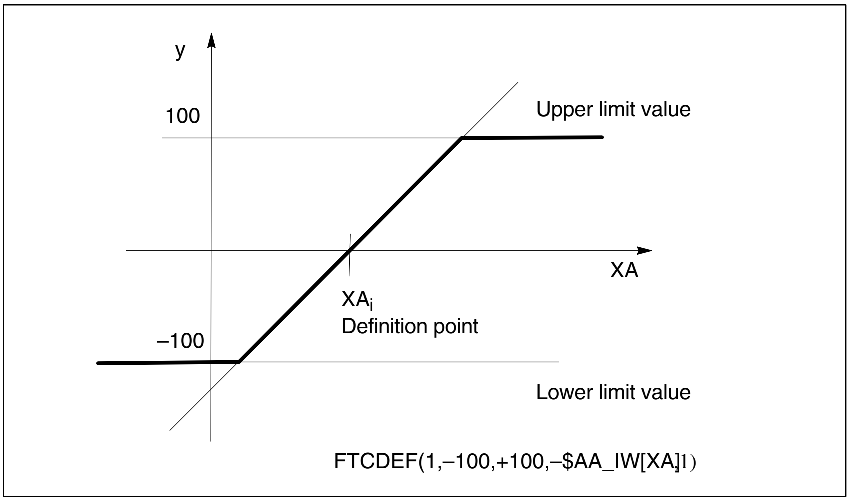



Fig. 2-12 Straight line with gradient 1

### 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 to which correction value is added |
| Channel no.: | Channel in which offset must take effect |
| Spindle no.: | Spindle to which offset must be applied |

The online tool offset is activated before the dresser axis movement block.

**Example:**

```
FCTDEF (1, –100, 100, –$AA_IW[X], 1)     ; Define function
PUTFTOCF (1, $AA_IW[X], 1, 2, 1)          ; Write online TO
                                          ; 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 there is ... | then ... |
|---|---|
| no channel no. is specified | the online offset will take effect in this channel |
| no spindle no. is specified | the online offset will be applied to the current tool |

---

With SW 3.2 and higher, an online tool offset can be called as a synchronized action.

**References:** /FB/, S5, "Synchronized Actions"

## 2.2.2 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 must be applied (machining channel for grinding operation).

FTOCOF always corresponds to reset position. PUTFTOC commands are effective only when the parts program and command FTOCON are active.

---

### 2.2.3 Example of writing online tool offset continuously

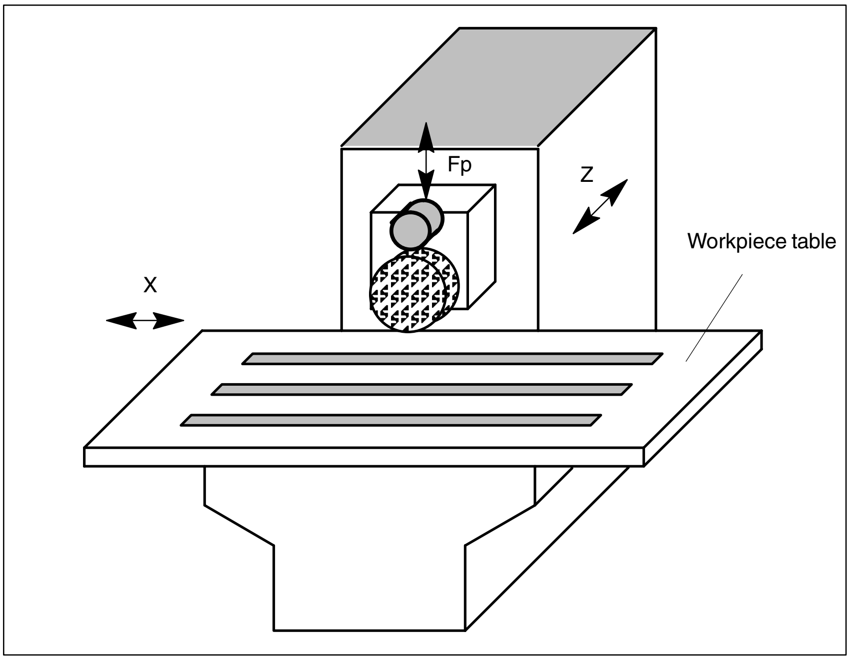



Fig. 2-13 Surface grinding machine:

Y: Infeed axis for grinding wheel
V: Infeed axis for dressing roller
X: Oscillation axis, left – right
Plane for 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**

```
...
G1 G19 F10 G90                      ;Basic setting
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)                           ;Start program in channel 2

Y200                                ;Traverse to target position
...
M30
```

**Dressing program in channel 2 _N_ABRICHT_MPF**

```
...
FCTDEF (1, –1000, 1000, –$AA_IW[V], 1)
                                ;Define function
PUTFTOCF (1, $AA_IW[V], 2, 1)   ;Write online tool offset continuously
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.

## 2.2.4 Write online tool offset discretely

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 there is ... | then ... |
|---|---|
| no channel no. is specified | the online offset will take effect in this channel |
| no spindle no. is specified | the online offset will be applied to the current tool |

### 2.2.5 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, but only 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 overlaid on the programmed movement of the axis. The programmed limit values (e.g. speed) are taken into account.
  If a DRF offset and online tool offset are both activated for an axis, the DRF offset has higher priority and is applied first.
- The valid offset is traversed at JOG velocity allowing for the specified maximum acceleration rate.
  Channel-specific MD20610: ADD_MOVE_ACCEL_RESERVE is taken into account with respect to FTOCON. 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.

## 2.3 Online tool radius compensation

**General**

When the longitudinal axis of the tool and the contour are mutually perpendicular, the offset quantity can be applied as a length compensation to one of the three geometry axes (online tool length compensation, see Section 2.2).

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 compensation function is enabled via MD 20254: $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, see Subsection 2.2.2).

**Parameterization**

The parameters of the online tool offset are set by means of commands PUTFTOCF (see Subsection 2.2.1) and PUTFTOC (see Subsection 2.2.4). 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.
- Application of online tool radius compensation

  On 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.

## 2.4 Grinding-specific tool monitoring

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 parts program or
- automatically through selection of tool length compensation of a grinding tool with uneven tool type number.

---

**Note**

Automatic selection of the monitor must be set via channel-specific MD20350: 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.

---

### 2.4.1 Geometry monitoring

The following quantities can be monitored

- The current grinding wheel radius
- The current grinding wheel width

The current wheel radius is compared to the value stored in parameter $TC_TPG3 (see Subsection 2.1.3).
The current radius is compared to 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 DB31–48 at the PLC interface.
This bit is otherwise set to “0”.

DB31–48, DBX83.3 = 1 ⇒ Geometry monitor has responded
DB31–48, DBX83.3 = 0 ⇒ Geometry monitor has not responded

**Note**

No error reaction is initiated internally in the control system.

### 2.4.2 Speed monitoring

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 $min^{-1}$.

The monitoring function operates cyclically and is designed to react to the first limit value reached.

**When does monitoring take place?**

Monitoring of the speed for violation of the limit value takes place cyclically, allowing for the spindle speed override.

**When is the speed limit value reset?**

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
- IS “Speed monitoring” (DB31–48, DBX83.6) is output.

DB31–48, DBX83.6 = 1 ⇒ Limit value of speed monitoring reached
DB31–48, DBX83.6 = 0 ⇒ Limit value of speed monitoring not reached

---

**Note**

No error reaction is initiated internally in the control system.

---

## 2.4.3 Selection/deselection of tool monitoring

The following parts program commands are provided for selecting and deselecting the grinding-specific tool monitor of an active or inactive tool:

| command | Meaning |
|---|---|
| **TMON**<br>Tool monitoring on | Selection of tool monitoring for the active tool in the channel |
| **TMOF**<br>tool monitoring Off | Deselection of tool monitoring for the active tool in the channel |
| **TMON (T number)**<br>tool monitoring on (t no.) | Selection of tool monitoring for a non-active tool with T number |
| **TMOF (T number)**<br>tool monitoring off (t no.) | Deselection of tool monitoring for a non-active tool with T number |
| **TMOF (0)**<br>tool monitoring off (0) | Deselection of tool monitoring for all tools |

## 2.5 Constant grinding wheel peripheral speed (GWPS)

**What is GWPS?**

The grinding wheel peripheral speed is generally programmed for grinding wheels rather than a spindle speed. This is a quantity that is determined by the technological process (e.g. grinding wheel characteristic, 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:

$$n[rev/^{min}] = \frac{GWPS[m*s^{-1}] * 60}{2\pi * R[m]}$$

---

**Note**

- The grinding wheel peripheral speed can be programmed and selected for grinding tool types (400 to 499).
  Wear is taken into account in calculating the radius (parameter $TC_TPG9).
- This function also applies to inclined wheels/axes.
- The relevant wear and the tool base dimension (as a function of tool type) are added to the parameter selected by $TC_TPG9. The product is divided by cos ($TC_TPG8) if parameter $TC_TPG8 (angle of inclined grinding wheel) is positive and by sin ($TC_TPG8) if it 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).

### 2.5.1 Selection/deselection and programming of GWPS, system variable

The GWPS is selected and deselected with the following parts program commands:

| Command | Meaning |
|---|---|
| **GWPSON**<br>Grinding wheel peripheral speed On | Selection of GWPS for the active tool in channel |
| **GWPSOF**<br>grinding wheel peripheral speed Off | Deselection of GWPS for the active tool in channel |
| **GWPSON(T number)**<br>Grinding wheel peripheral speed on (t no.) | Selection of GWPS for a non-active tool with T number |
| **GWPSOF(T number)**<br>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.<br>Unit of value setting depends on basic system (m/s or ft/s). |

**References:** /PA/, Programming Guide

**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 must 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 before it can be activated again with the new tool (otherwise an alarm is output).
- 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 in the part program to determine whether GWPS is active for a specific spindle.
TRUE: GWPS programming of spindle active
FALSE: GWPS programming of spindle not active

**References:** /PG/, Programming Guide

### 2.5.2 GWPS in all operating modes

**General**

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 parts program end.

The function is activated via MD 35032: $MA_SPIND_FUNC_RESET_MODE (parameterization of GWPS function).

**GWPS after POWER ON**

A grinding-specific tool is defined via the following MD:
MD 20110: $MC_RESET_MODE_MASK
MD 20120: $MC_TOOL_RESET_VALUE
MD 20130: $MC_CUTTING_EDGE_RESET_VALUE

| If | and | then |
|---|---|---|
| MD 35032: $MA_SPIND_FUNC_RESET_MODE is set | the tool is a grinding-specific tool type (400 to 499, MD 20110, 20120, 20130) with reference to a valid spindle (parameter $TC_TPG1), | GWPS is activated for this spindle. |
| **Note:** GWPS is deselected for all other spindles in this channel. | | |

**GWPS after RESET/parts program end**

After a RESET/parts program end, GWPS remains active for all spindles for which it was already selected.

| If | and | then |
|---|---|---|
| MD 35032: $MA_SPIND_FUNC_RESET_MODE is set | GWPS is active on RESET or parts program end, | GWPS remains active for this spindle. |
| MD 35032: $MA_SPIND_FUNC_RESET_MODE is not set | GWPS is active on RESET or parts program end, | GWPS is deactivated for this spindle. |
| **Note:** GWPS is deselected for all other spindles in this channel. | | |

Via the MD 35040: $MA_SPIND_ACTIVE_AFTER_RESET can be set to determine whether the spindle must continue to rotate at the current speed 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 parts program/overstoring
- programming the grinding wheel peripheral speed through assignment to address “S” in MDA
- spindle speed control via PLC (FC18).

**“GWPS active” interface signal**

IS “GWPS active” (DB31, ... , DBX84.0) indicates whether or not the GWPS function is active.

### 2.5.3 Example of how to program 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**

```
N20  T1 D1                  ;Select T1 and D1
N25  S1=1000 M1=3           ;1000 rev/min for spindle 1
N30  S2=1500 M2=3           ;1500 rev/min for spindle 2
...
N40  GWPSON                 ;GWPS selection for active tool T1
N45  S[$P_AGT[1]] = 60      ;Set GWPS for active tool to 60 m/s
                              n=1909.85 min–1
....
N50  GWPSON(5)              ;GWPS selection for tool 5 (2nd spindle)
N55  S[$TC_TPG1[5] ] = 40   ;Set GWPS for spindle 2 to 40 m/s
                              n=1909.85 min–1
...
N60  GWPSOF                 ;Switch off GWPS for active tool
N65  GWPSOF(5)              ;Switch off GWPS for tool 5 (spindle 2)
...
```

---

**Note**

Programming of tool monitoring function See Section 2.4.

---

**Supplementary references**

References: /FB/, P5, Oscillation
/FB/, V1, Feeds, Multiple Feeds in a Block
/FB/, S5, Synchronized Actions

■

## Notes

# Supplementary Conditions 3

**Grinding-specific tool offset with grinding wheel peripheral speed**

This function is available for

- SINUMERIK 810D/810D SW2 and higher

**Continuous dressing (parallel dressing)**

The function is an option and is available for

- SINUMERIK 840D with NCU 572/573, SW 2 and higher

The function is contained in the export version 840DE with restricted functionality; it is not contained in the 810DE (up to SW 3.1).

- The function is contained in the export version 810DE with restricted functionality in SW 3.2 and higher.

**Tool changes with online tool offset for grinding operations**

Tool changes can be executed in conjunction with the online tool offset through the selection of T numbers on the

- SINUMERIK 810D with CCU1/CCU2, SW 3 and higher
- SINUMERIK 840D with NCU 572/573, SW 5 and higher

The option of changing tools with M6 in conjunction with the online tool offset has not be implemented in the currently available software versions.

■

# Notes

# Data Descriptions (MD, SD) 4

## 4.1 Channelspecific machine data

| | | | | | |
|---|---|---|---|---|---|
| **20254**<br>MD number | **ONLINE_CUTCOM_ENABLE**<br>Enable online tool radius compensation | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: 2 | | Unit: – |
| Data type: BOOLEAN | | | | Applies from SW: 4.1 | |
| Meaning: | This data enables online tool radius compensation.<br>When the function is enabled, the control reserves the necessary memory space required for online tool radius compensation after POWER ON.<br><br>ONLINE_CUTCOM_ENABLE = 0: Online tool radius compensation can be used<br>ONLINE_CUTCOM_ENABLE = 1: Online tool radius compensation cannot be used | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **20350**<br>MD number | **TOOL_GRIND_AUTO_TMON**<br>Automatic tool monitoring | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 1 | |
| Changes effective after POWER ON | | | Protection level: | | Unit: – |
| Data type: BYTE | | | | Applies from SW: 2.1 | |
| Meaning: | This MD defines whether the tool monitoring function is automatically activated when the tool length compensation of a grinding tool with monitoring is selected.<br>TOOL_GRIND_AUTO_TMON = 1 : Automatic monitoring activated<br>TOOL_GRIND_AUTO_TMON = 0 : Automatic monitoring deactivated | | | | |

## 4.2 Axis-specific machine data

| | | | | | |
|---|---|---|---|---|---|
| **35032**<br>MD number | **SPIND_FUNC_RESET_MODE**<br>Parameterization of GWPS function | | | | |
| Default setting: 0 | | Minimum input limit: 0 | | Maximum input limit: 0x01 | |
| Changes effective after POWER ON | | | Protection level: | | Unit: – |
| Data type: DWORD | | | | Applies from SW: 4.1 | |
| Meaning: | This data allows the “GWPS in every operating mode” function to be selected/deselected.<br><br>SPIND_FUNC_RESET_MODE, Bit 0 = 0 : “GWPS in every operating mode” is deselected<br>SPIND_FUNC_RESET_MODE, Bit 0 = 1 : “GWPS in every operating mode” is selected | | | | |

■

# Notes

# 5 Signal Descriptions

| DB31, ... **DBX83.3** Data Block | **Geometry monitoring** Signal(s) NCK → PLC | |
|---|---|---|
| Edge evaluation: – | Signal(s) updated: – | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | Error in grinding wheel geometry<br><br>**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 signal transition 1 ––> 0 | No error in grinding wheel geometry | |
| Application example(s) | Grinding-specific tool monitoring | |
| References | See Section 2.4 | |

| DB31, ... **DBX83.6** Data Block | **Speed monitoring** Signal(s) NCK → PLC | |
|---|---|---|
| Edge evaluation: – | Signal(s) updated: – | Signal(s) valid from SW: 2.1 |
| Signal state 1 or signal transition 0 ––> 1 | Error in grinding wheel speed<br><br>**Note:** No further reaction to this signal state is programmed. Reactions deemed necessary must be programmed by the PLC user. | |
| Signal state 0 or signal transition 1 ––> 0 | No error in grinding wheel speed | |
| Application example(s) | Grinding-specific tool monitoring | |
| References | See Section 2.4 | |

| DB31, ... **DBX84.1** Data Block | **GWPS active** Signal(s) NCK → PLC | |
|---|---|---|
| Edge evaluation: – | Signal(s) updated: – | Signal(s) valid from SW: 4.1 |
| Signal state 1 or signal transition 0 ––> 1 | Constant grinding wheel peripheral speed (GWPS) is active.<br><br>If GWPS is active, then all S value inputs from the PLC are interpreted as the grinding wheel peripheral speed. | |
| Signal state 0 or signal transition 1 ––> 0 | Constant grinding wheel peripheral speed (GWPS) is not active. | |
| Application example(s) | GWPS in all operating modes. | |
| References | See Subsection 2.5.2 | |

■

# 6 Example

None

■

## Notes

# Data Fields, Lists 7

## 7.1 Interface signals

| DB number | Bit, byte | Name | Reference |
|---|---|---|---|
| Axis/spindle-specific | | | |
| 31, ... | 83.3 | Geometry monitoring (SW 2 and higher) | |
| 31, ... | 83.6 | Speed monitoring (SW 2 and higher) | |
| 31, ... | 84.1 | GWPS active (SW 4 and higher) | |

## 7.2 Machine data

| Number | Names | Name | Reference |
|---|---|---|---|
| General ($MN_ ... ) | | | |
| 18094 | MM_NUM_CC_TDA_PARAM | Number of TOA data | /FBW/<br>/S7/ |
| 18096 | MM_NUM_CC_TOA_PARAM | Number of TOA data which can be set up per tool and evaluated by the CC | /FBW/<br>/S7/ |
| 18100 | MM_NUM_CUTTING_EDGES_IN_TOA | Tool offsets per TOA module | S7 |
| Channelspecific ($MC_ ... ) | | | |
| 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 | K1 |
| Axis-specific ($MA_ ... ) | | | |
| 32020 | JOG_VELO | Conventional axis velocity | H1 |
| 35032 | SPIND_FUNC_RESET_MODE | Parameterization of GWPS function | |

## 7.3 Interrupts

A more detailed description of the alarms which may occur is given in
**References:** /DA/, Diagnostic Guide
or in the online help in systems with MMC 101–103.

■

## Notes

# Index

## Symbols



## Numbers



## A



## B



## C

## D



## E



## F

## G



## H



## I



## J



## K

## L



## M



## N

## O

## P



## Q



## R

## S

## T



## U



## V



## W

| | |
|---|---|
| To<br>SIEMENS AG<br>A&D MC BMS<br>P.O. Box 3180<br>D-91050 Erlangen, Germany<br>(Tel. +49 (0) 180 / 5050-222 [Hotline]<br>Fax +49 (0) 9131 / 98-2176 [Documentation]<br>E-mail: motioncontrol.docu@erlf.siemens.de) | **Recommendations**<br>**Corrections**<br>For Publication/Manual:<br>SINUMERIK 840D/840Di/810D<br>Extended Functions (Part 2)<br>Order No.: 6FC5297-7AC30-0BP1<br>Edition: 10.04<br>Manufacturer/Service Documentation |
| **From**<br>Name<br>Your company address/Dept.<br>Address<br>Zip code: City:<br>Phone: /<br>Fax: / | Description of Functions<br><br>Should you come across any printing errors when reading this publication, please notify us on this sheet.<br>Suggestions for improvement are also welcome. |

**Recommendations and/or corrections**

# Overview of SINUMERIK 840D/840Di/810D Documentation (10.2004)

## General Documentation

- *SINUMERIK* 840D/810D — Brochure
- *SINUMERIK* 840D/840Di/ 810D/ — Catalog Ordering Info. NC 60 *)
- *Safety Integrated* Application Manual — **Safety Integrated** Application Manual

## User Documentation

- *SINUMERIK* 840D/810D/ FM-NC — **AutoTurn**
  – Short Guide
  – Programming/ Setup
- *SINUMERIK* 840D/840Di/ 810D — Operator's Guide **– HT 6**
- *SINUMERIK* 840D/840Di/ 810D — **Diagnostics Guide *)**
- *SINUMERIK* 840D/840Di/ 810D — **Operator's Guide *)**
  – Short Guide
  – HMI Embedded
  – HMI Advanced

## User Documentation

- *SINUMERIK* 840D/840Di/ 810D — **Programming Guide**
  – Short Guide
  – Fundamentals *)
  – Advanced *)
  – Cycles
  – Measuring Cycles
  – ISO Turning/Milling
  – Lists **System Variables**
- *SINUMERIK* 840D/810D — Operator's Guide
  – **ManualTurn**
  – Short Guide ManualTurn
  – **ShopMill**
  – Short Guide ShopMill
  – **ShopTurn**
  – Short Guide ShopTurn
- *SINUMERIK* 840Di — **System Overview**

## Manufacturer/Service Documentation

- *SINUMERIK* — **Configuring** (HW) *)
  – 810D
  – 840D
- *SINUMERIK* 840D/840Di/ 810D — **Operator Components** (HW) *)
- *SINUMERIK* 840D/840Di/ 810D — Description of Functions
  **– ManualTurn**
  **– ShopMill**
  **– ShopTurn**

## Manufacturer/Service Documentation

- *SINUMERIK* 611D 840D/810D — Description of Functions **Synchronized Actions**
- *SINUMERIK SIMODRIVE* 840D/840Di/ 810D — Description of Functions **Drive Functions *)**
- *SINUMERIK* 840D/840Di/ 810D — **Description of Functions**
  – Basic Machine *)
  – Extended Functions
  – Special Functions
- *SINUMERIK* 840D/810D — Description of Functions **Tool Management**
- *SINUMERIK* 840D/810D — **Configuring Kit** HMI Embedded
- *SINUMERIK* 840D/810D — Description of Functions Operator Interface **OP 030**
- *SINUMERIK* 840D/810D — **MCIS**
  – Computer Link
  – Tool Data Inform. Syst.
  – NC Data Management
  – NC Data Transfer
  – Tool Data Communication

## Manufacturer/Service Documentation

- *SINUMERIK SIMODRIVE* — Description of Functions SINUMERIK **Safety Integrated**
- *SINUMERIK* 840D — Description of Functions **Digitizing**
- *SINUMERIK SIMODRIVE* — **Installation & Start-Up Guide *)**
  – 810D
  – 840D/611D
  – HMI
- *SINUMERIK SIMODRIVE* 840D/840Di 810D 611D — **Lists *)**
- *SINUMERIK SIMODRIVE* 840D 611D — Description of Functions **Linear Motor**
- *SINUMERIK SIMODRIVE* 840D 611D — Description of Functions
  **– Hydraulics Module**
  **– Analog Module**
- *MOTION CONTROL SYSTEMS* — **EMC** Guidelines

## Electronic Documentation

- *SINUMERIK SIMODRIVE* 840D/840Di/ 810D 611, Motors — **DOC ON CD *)** The SINUMERIK System

## Manufacturer/Service Documentation

- *SINUMERIK* 840D/840Di/ 810D — Description of Functions **ISO Dialects** for SINUMERIK
- *SINUMERIK* 840Di — **Manual** (HW + Installation and Start-Up)
- *SINUMERIK* 840D/810D — Description of Functions **Remote Diagnosis**
- *SINUMERIK* 840D/840Di/ 810D — **Manual @ Event**

---

*) These documents are a minimum requirement

**Siemens AG**
Automation & Drives
Motion Control Systems
P. O. Box 3180, D – 91050 Erlangen
Germany

www.siemens.com/motioncontrol


Order No.: 6FC5297-7AC30-0BP1

Printed in Germany