SIEMENS

SINUMERIK

SINUMERIK 840D sl / 828D Basic Functions

Function Manual

Valid for

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

Software version
CNC software 4.5 SP1

07/2012
6FC5397-0BP40-3BA0

Legal information

Warning notice system

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

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

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

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

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

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-0BP40-3BA0
℗ 08/2012 Technical data subject to change

Preface

SINUMERIK documentation

The SINUMERIK documentation is organized in the following categories:

- General documentation
- User documentation
- Manufacturer/service documentation

Additional information

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

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

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

docu.motioncontrol@siemens.com

My Documentation Manager (MDM)

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

www.siemens.com/mdm

Training

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

- www.siemens.com/sitrain

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

  SinuTrain - training software for SINUMERIK

FAQs

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

SINUMERIK

You can find information on SINUMERIK under the following link:

www.siemens.com/sinumerik

Target group

This publication is intended for:

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

Benefits

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

Standard version

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

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

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

Technical Support

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

Information on the structure and contents

Installation

Structure of this Function Manual:

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

Note

For detailed descriptions of data and alarms see:

- For machine and setting data:
  Detailed description of machine data (only electronically on DOConCD or DOConWEB)
- For NC/PLC interface signals:
  - Function Manual, Basic Functions; NC/PLC Interface Signals (Z1)
  - Function Manual, Basic Functions; NC/PLC Interface Signals (Z2)
  - Function Manual, Special Functions; NC/PLC Interface Signals (Z3)
- For alarms:
  Diagnostics Manual

Notation of system data

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

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

Note

Signal address

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

Quantity structure

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

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

Data types

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

Type	Meaning	Range of values
INT	Signed integers	-2147483648 ... +2147483647
REAL	Figures with decimal point acc. to IEEE	$\pm(2.2*10^{-308} \ldots 1.8*10^{+308})$
BOOL	Truth values TRUE (1) and FALSE (0)	1, 0
CHAR	ASCII characters	Corresponding to code 0 to 255
STRING	Character string, number of characters in [...]	Maximum of 200 characters (no special characters)
AXIS	Axis names only	All axis identifiers in the channel
FRAME	Geometrical parameters for moving, rotating, scaling, and mirroring	
Arrays can only be formed from similar elementary data types. Up to 3-dimensional arrays are possible.		

Table of Contents

A2: Various NC/PLC interface signals and functions 1

1.1 Brief description

Content

The PLC/NCK interface comprises a data interface on one side and a function interface on the other. The data interface contains status and control signals, auxiliary functions and G functions, while the function interface is used to transfer jobs from the PLC to the NCK.

This Description describes the functionality of interface signals, which are of general relevance but are not included in the Descriptions of Functions.

- Asynchronous events
- Status signals
- PLC variable (read and write)

1.2 NC/PLC interface signals

1.2.1 General

NC/PLC interface

The NC/PLC interface comprises the following parts:

- Data interface
- Function interface

Data interface

The data interface is used for component coordination:

- PLC user program
- NC
- HMI (operator control component)
- MCP (machine control panel)

Data exchange is organized by the basic PLC program.

Cyclic signal exchange

The following interface signals are transferred cyclically, i.e. in the clock grid of the OB1, by the basic PLC program:

- NC and operator-panel-front-specific signals
- Mode group-specific signals
- Channel-specific signals
- Axis/spindle-specific signals

NC and operator-panel-front-specific signals (DB10)

PLC to NC:

- Signals for influencing the CNC inputs and outputs
- Keyswitch signals (and password)

NC to PLC:

- Actual values of CNC inputs
- Setpoints of CNC outputs
- Ready signals from NC and HMI
- NC status signals (alarm signals)

Channel-specific signals (DB21, ...)

PLC to NC:

- Control signal "Delete distance-to-go"

NC to PLC:

- NC status signals (NCK alarm active)

Axis/spindle-specific signals (DB31, etc.)

PLC to NC:

- Control signals to axis/spindle (e.g. follow-up mode, servo enable, etc.)
- Control signals to drive (bytes 20, 21)

NC to PLC:

- Status signals from axis/spindle (e.g. position controller active, current controller active, etc.)
- Control signals from drive (bytes 93, 94)

Function interface

The function interface is generated by function blocks (FB) and function calls (FC). Function requests, e.g. to traverse axes, are sent from the PLC to the NC via the function interface.

References

- Description of the basic PLC program:

  → Function Manual, Basic Functions, Basic PLC Program (P3)
- Description of the event-driven signal exchange (auxiliary and G functions):

  → Function Manual, Basic Functions; Auxiliary Function Output to PLC (H2)
- Overview of all interface signals, functional and data components:

  → List Manual 2

1.2.2 Ready signal to PLC

DB10 DBX104.7 (NCK CPU ready)

The NCK CPU is ready and registers itself cyclically with the PLC.

DB10 DBX108.3 (HMI - CPU at OPI ready)

SINUMERIK Operate is ready and registers itself cyclically with the NC.

DB10 DBX108.5 (drives in cyclic operation)

Requirement: For **all** machine axes of the NC, the associated drives are in the cyclic operation, i.e. they cyclically exchange PROFIdrive telegrams with the NC.

DB10 DBX108.6 (drive ready)

Requirement: For **all** machine axes of the NC, the associated drives and also the third-party drives are ready via PROFIBUS:

DB31, ... DBX93.5 == 1 (drive ready)

DB10 DBX108.7 (NC ready)

The NC is ready.

1.2.3 Status signals to PLC

DB10 DBX103.0 (remote diagnosis active)

The HMI component reports to the PLC that the remote diagnostics (option) is active, i.e. controlling is done via an external PC.

DB10 DBX109.6 (ambient temperature alarm)

The ambient temperature or fan monitoring function has responded.

DB10 DBX109.7 (NCK battery alarm)

The battery voltage has dropped below the lower limit value. The control can still be operated. A control system shutdown or failure of the supply voltage will lead to loss of data.

DB10 DBX109.0 (NCK alarm pending)

The NC signals that at least one NC alarm is pending. The channel-specific interface can be scanned to see which channels are involved and whether this will cause a processing stop.

DB21, ... DBX36.6 (channel-specific NCK alarm pending)

The NC sends this signal to the PLC to indicate that at least one NC alarm is pending for the relevant channel (see also DB21, ... DBX36.7).

DB21, ... DBX36.7 (NCK alarm with processing stop present)

The NC sends this signal to the PLC to indicate that at least one NCK alarm which has interrupted or aborted the current program run (processing stop) is pending for the affected channel.

1.2.4 Signals to/from the operator panel front

DB19 DBX0.0 (screen bright)

The screen blanking is disabled.

DB19 DBX0.1 (screen dark)

The operator panel screen is darkened.

If the interface signal is used to actively darken the screen:

- It is no longer possible to switch the screen bright again on the keyboard (see below).
- The first keystroke on the operator panel front already triggers an operator action.

Note

In order to prevent accidental operator actions when the screen is darkened via the interface signal, we recommend disabling the keyboard **at the same time**.

DB19 DBX0.1 = 1 **AND** DB19 DBX0.2 = 1 (key disable)

Screen darkening via keyboard/automatic screen saver

If no buttons are pressed on the operator panel front within the assigned time (default = 3 minutes):
MD9006 $MM_DISPLAY_BLACK_TIME
(time for screen darkening), the screen is automatically darkened.

The screen lights up again the first time a button is pressed following darkening. Pressing a button to lighten the screen will not generate an operator action.

Parameterization:

DB19 DBX0.1 = 0

MD9006 $MM_DISPLAY_BLACK_TIME > 0

DB19 DBX0.2 (key disable)

All inputs via the connected keyboard are inhibited.

DB19 DBX 0.3 / 0.4 (Delete cancel alarms / Delete recall alarms) (HMI Advanced)

Request to delete all currently pending alarms with Cancel or Recall delete criterion. Deletion of the alarms is acknowledged via the following interface signals.

- DB19 DBX20.3 (cancel alarm deleted)
- DB19 DBX20.4 (recall alarm deleted)

DB19 DBX0.7 (actual values in WCS, 0=MCS)

Switching over of actual-value display between machine and workpiece coordinate system:

- DB19 DBX0.7 = 0: Machine coordinate system (MCS)
- DB19 DBX0.7 = 1: Workpiece coordinate system (WCS)

DB19 DBB13 (control of the file transfer via hard disk) (HMI Advanced only)

Job byte to control file transfer via hard disk. The jobs relate to the user control file in the interface signals:

DB19 DBB16 (part program handling: Number of the control file for user file names)

DB19 DBB17 (part program handling: Index of the file to be transmitted from the user list)

DB19 DBB16 (control of file transfer via hard disk) (HMI Advanced only)

Control byte for file transfer via hard disk to define the index for the control file (job list). This file is handled according to the job in the interface signal:

DB19 DBB13

DB19 DBB17 (part program handling: Index of the file to be transferred from the user list)

Control byte for file transfer via hard disk to indicate the line in the user control file in which the control file to be transferred is stored

DB19 DBB26 (part program handling: Status)

Status byte for current status of data transfer for "select", "load" or "unload", or if an error occurred during data transmission.

DB19 DBB27 (error program handling)

Output byte for error values for data transfer via hard disk.

1.2.5 Signals to channel

DB21, ... DBX6.2 (delete distance-to-go)

The rising edge on the interface signal generates a stop on the programmed path in the corresponding NC channel with the currently active path acceleration. The path distance-to-go is then deleted and the block change to the next part-program block is enabled.

1.2.6 Signals to axis/spindle

DB31, ... DBX1.0 (drive test travel enable)

If machine axes are traversed by special test functions such as "function generator", an explicit drive-test-specific enable is requested for the motion:

DB31, ... DBX61.0 = 1 (drive test travel request)

The motion is carried out once the motion is enabled:

DB31, ... DBX1.0 == 1 (drive test travel enable)

NOTICE
It is the sole responsibility of the machine manufacturer / system startup engineer to take suitable action/carry out appropriate tests to ensure that the machine axis can be traversed during the drive test without putting personnel or machinery at risk.

DB31, ... DBX1.3 (axis/spindle disable)

Axis disable when machine axis is at rest

No traversing request (manual or automatic) is carried out for a machine axis at rest and NC/PLC interface signal:

DB31, ... DBX1.3 == 1 (axis/spindle disable)

The traversing request is maintained. If the axis disable is canceled when a traversing request is pending DB31, ... DBX1.3 = 0 the motion is carried out.

Axis disable when machine axis in motion

When machine axis is in motion and NC/PLC interface signal DB31, ... DBX1.3 == 1 the motion of the machine axis is decelerated to a standstill via the axis-specific brake characteristics currently active or, if it is part of an interpolated path motion or coupling, it is decelerated on a path or coupling-specific basis.

The motion is continued if the axis disable is canceled by another pending traversing request: DB31, ... DBX1.3 = 0.

Spindle disable

The response is determined by the current spindle mode:

- Control mode: Speed setpoint = 0
- Positioning mode: See above "Axis disable".

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

"Follow-up mode" is only effective in conjunction with the NC/PLC interface signal:

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

DB31, ... DBX2.1	DB31, ... DBX1.4	Function
1	Ineffective	Normal operation (machine axis in closed-loop control mode)
0	1	Follow-up
0	0	Hold

Function: Follow-up

During follow-up, the setpoint position of the machine axis is continuously corrected to the actual position (setpoint position = actual position).

The following interface signals have to be set for the follow-up function:

DB31, ... DBX2.1 = 0 (controller enable)

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

Feedback:

DB31, ... DBX61.3 = 1 (follow-up active)

Note

When the controller enable is set from follow-up mode, if the part program is active, the last programmed position is approached again internally in the NC (REPOSA: Approach along line on all axes). In all other cases, all subsequent motions start at the current actual position.

During "follow-up", clamping or zero-speed monitoring are **not active**.

Function: Hold

The hold function does not correct the setpoint position of the machine axis to the actual position. If the machine axis moves away from the setpoint position, a following error (difference between setpoint and actual position) is generated. This error is corrected "suddenly" when the controller enable is set by the position controller, without observing the axial acceleration characteristic.

The following interface signals have to be set for the hold function:

DB31, ... DBX2.1 = 0 (controller enable)

DB31, ... DBX1.4 = 0 (follow-up mode)

Feedback:

DB31, ... DBX61.3 = 0 (follow-up active)

During "hold", clamping or zero-speed monitoring are **active**.

NOTICE
With the "hold" function, once the controller enable has been set, the setpoint/actual-value difference is corrected directly by the position controller, i.e. without following the axial acceleration characteristic.

Application example

Positioning response of machine axis Y following clamping when "controller enable" set. Clamping pushed the machine axis from the actual position Y_1 to the clamping position Y_k.

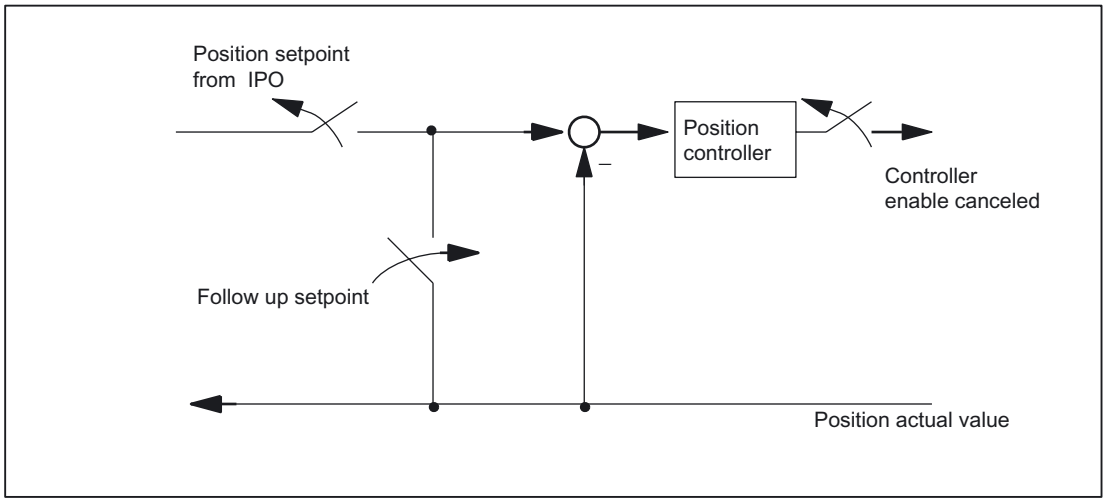


Figure 1-1 Effect of controller enable and follow-up mode

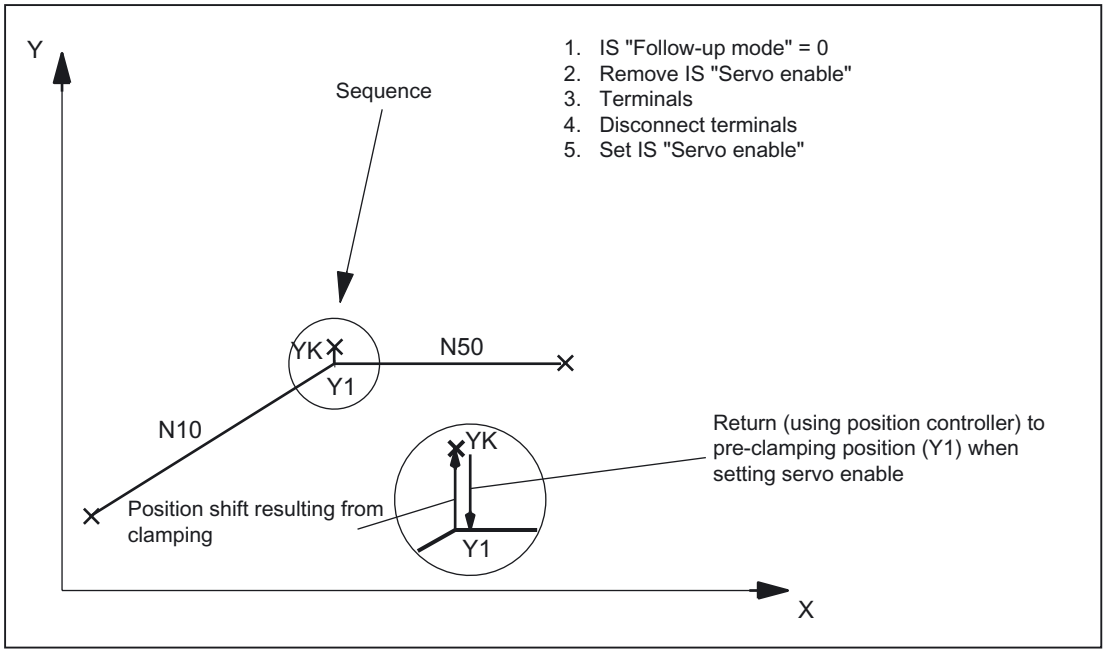


Figure 1-2 Trajectory for clamping and "hold"

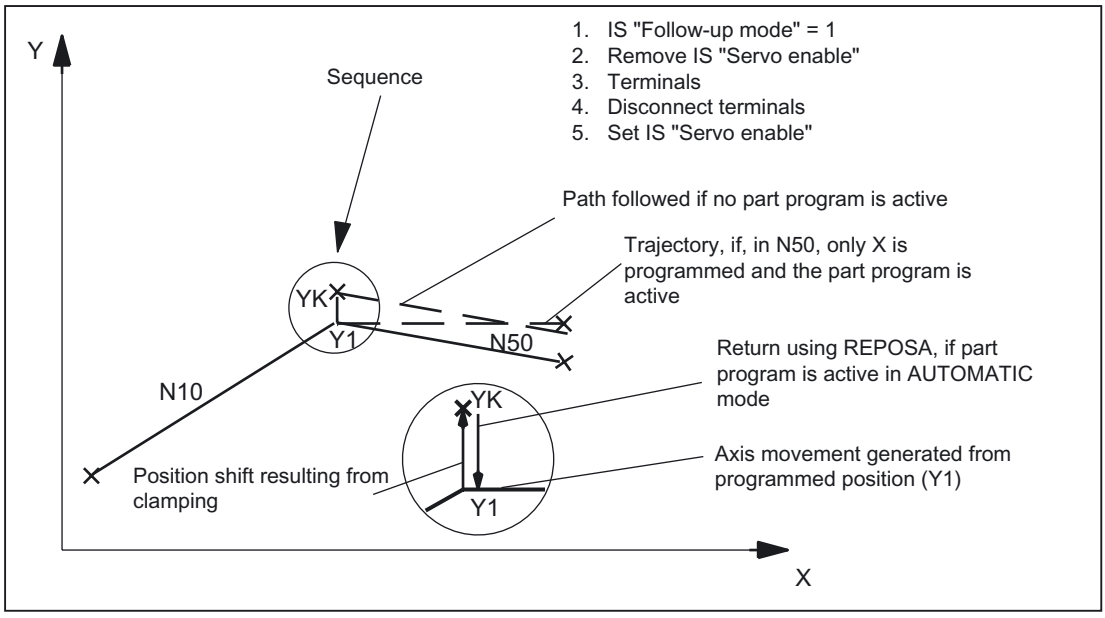


Figure 1-3 Trajectory for clamping and "follow-up"

Drives with analog setpoint interface

A drive with an analog setpoint interface is capable of traversing the machine axis with an external setpoint. If "follow-up mode" is set for the machine axis, the actual position continues to be acquired. Once follow-up mode has been canceled, referencing is not required.

The following procedure is recommended:

1. Activate follow-up mode:

   DB31, ... DBX2.1 = 0 (controller enable)

   DB31, ... DBX1.4 = 1 (follow-up mode) (in the same or preceding OB1 cycle)

   → The axis/spindle is operating in follow-up mode

2. Deactivate external controller enable and external speed setpoint

   → Axis/spindle moves with external setpoint

   → NC continues to detect the actual position and corrects the setpoint position to the actual position

3. Deactivate external controller enable and cancel external speed setpoint

   → Axis/spindle stops

4. Canceling follow-up mode

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

   DB31, ... DBX1.4 = 0 (follow-up mode)

   → NC synchronizes to current actual position. The next traversing motion begins at this position.

Note

"Follow-up mode" does not have to be canceled because it only has an effect in combination with "controller enable".

Canceling follow-up mode

Once follow-up mode has been canceled, the machine axis does not have to be referenced again if the maximum permissible encoder limit frequency of the active measuring system was not exceeded during follow-up mode. If the encoder limit frequency is exceeded, the controller will detect this:

- DB31, ... DBX60.4 / 60.5 = 0 (referenced/synchronized 1 / 2)
- Alarm: "21610 Encoder frequency exceeded"

Note

If "follow-up mode" is deactivated for a machine axis, which is part of an active transformation (e.g. TRANSMIT), this can generate motions as part of repositioning (REPOS) other machine axes involved in the transformation.

Monitoring

If a machine axis is in follow-up mode, the following monitoring mechanisms will not act:

- Zero-speed monitoring
- Clamping monitoring
- Positioning monitoring

Effects on other interface signals:

- DB31, ... DBX60.7 = 0 (position reached with exact stop fine)
- DB31, ... DBX60.6 = 0 (position reached with exact stop coarse)

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

Two measuring systems can be connected to one machine axis, e.g.

- Indirect motor measuring system
- Direct measuring system on load

Only one measuring system can be active at any one time. All closed-loop control, positioning operations, etc. involving the machine axis always relate to the active measuring system.

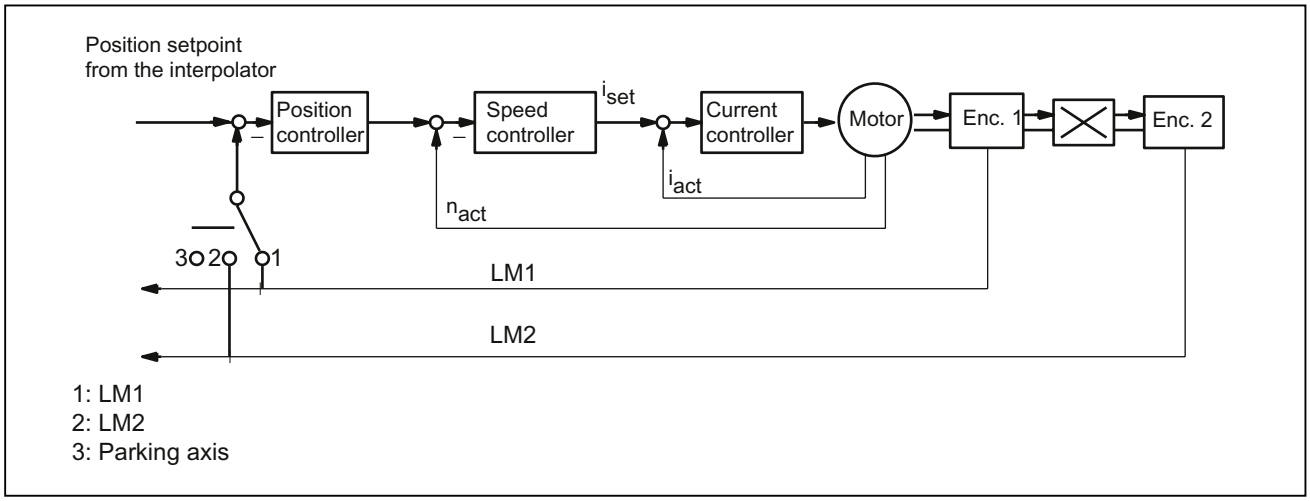


Figure 1-4 Position measuring systems 1 and 2

The table below shows the functionality of the interface signals in conjunction with the "controller enable".

DB31, ... DBX1.5	DB31, ... DBX1.6	DB31, ... DBX2.1	Function
1	0 (or 1)	1	Position measuring system 1 active
0	1	1	Position measuring system 2 active
0	0	0	"Parking" active
0	0	1	Spindle without position measuring system (speed-controlled)
1 -> 0	0 -> 1	1	Switchover: Position measuring system 1 → 2
0 -> 1	1 -> 0	1	Switchover: Position measuring system 2 → 1

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

Setting the controller enable closes the machine axis position control loop. The machine axis is in position control mode.

DB31, ... DBX2.1 == 1

Canceling the controller enable opens the machine axis position control loop and, subject to a delay, the machine axis speed control loop:

DB31, ... DBX2.1 == 0

Activation methods

The closed-loop controller enable for a machine axis is influenced by:

- NC/PLC interface signal:
  - DB31, ... DBX2.1 (controller enable)
  - DB31, ... DBX21.7 (pulse enable)
  - DB31, ... DBX93.5 (drive ready)
  - DB10, DBX56.1 (emergency stop)
- NCK-internal

  Alarms that trigger cancellation of the controller enable on the machine axes. The alarm responses are described in:

  References:
  Diagnostics Manual

Canceling the controller enable when the machine axis is at standstill:

- The machine axis position control loop opens
- DB31, ... DBX61.5 == 0 (position controller active)

Canceling the controller enable when the machine axis is in motion:

If a machine axis is part of an interpolatory path motion or coupling and the controller enable for this is canceled, all axes involved are stopped with a fast stop (speed setpoint = 0) and an alarm is displayed:

Alarm: "21612 Controller enable reset during motion"

- The machine axis is decelerated taking into account the parameterized duration of the braking ramp for error states with a fast stop (speed setpoint = 0):

  MD36610 $MA_AX_EMERGENCY_STOP_TIME (max. time duration of the braking ramp in event of errors)

  An alarm is displayed:

  Alarm: "21612 Controller enable reset during motion"

  Note

  The controller enable is canceled at the latest when the cutout time expires:

  MD36610 $MA_AX_EMERGENCY_STOP_TIME

- The machine axis position control loop opens. Feedback via interface signal:

  DB31, ... DBX61.5 == 0 (position controller active).

  The time for the parameterized cut-off delay of the controller enable is started by the machine data:

  MD36620 $MA_SERVO_DISABLE_DELAY_TIME (OFF delay of the controller enable)

- As soon as the actual speed has reached the zero speed range, the drive controller enable is canceled. Feedback via interface signal:

  DB31, ... DBX61.6 == 0 (speed controller active)

- The position actual value of the machine axis continues to be acquired by the controller.
- At the end of the braking operation, the machine axis is switched to follow-up mode, regardless of the corresponding NC/PLC interface signal. Zero-speed and clamping monitoring are not effective. See the description above for the interface signal:

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

Synchronizing the actual value (reference point approach)

Once the controller enable has been set, the actual position of the machine axis does not need to be synchronized again (reference point approach) if the maximum permissible limit frequency of the measuring system was not exceeded during the time in which the machine axis was not in position-control mode.

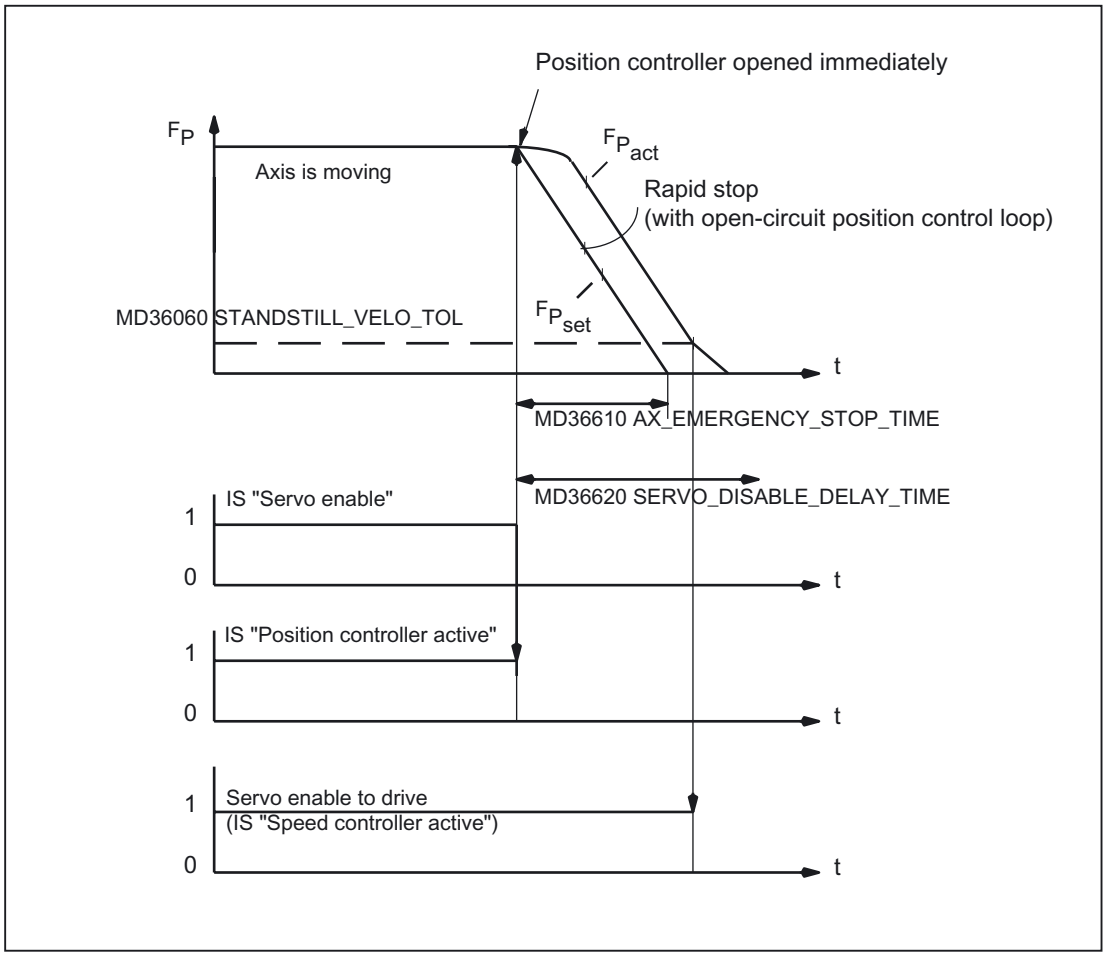


Figure 1-5 Canceling the controller enable when the machine axis is in motion

DB31, ... DBX2.2 (distance-to-go/spindle reset (axis/spindle-specific))

"Delete distance-to-go" is effective in AUTOMATIC and MDA modes only in conjunction with positioning axes. The positioning axis is decelerated to standstill following the current brake characteristic. The distance-to-go of the axis is deleted.

Spindle reset

For a detailed description of the spindle reset, see Section "S1: Spindles (Page 1383)".

DB31, ... DBX9.0 / 9.1 / 9.2 (controller parameter set)

Request for activation of the specified controller parameter set.

Controller parameter set	DBX9.2	DBX9.1	DBX9.0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Parameter set changeover must be enabled via the machine data (not required for spindles):

MD35590 $MA_PARAMSET_CHANGE_ENABLE = 1 or 2

For detailed information on the parameter set changeover, see Section "Parameter set selection during gear step change (Page 1437)".

Parameter set changeover when machine axis is in motion

The response to a parameter set changeover depends on the consequential change in the closed-loop control circuit gain factor Kv:

MD32200 $MA_POSCTRL_GAIN (servo gain factor) (servo gain factor)

- "Identical servo gain factors" or "position control not active":

  The NC responds immediately to the parameter set changeover. The parameter set is also changed during the motion.

- "Non-identical servo gain factors" or "position control active":

  In order to effect a changeover as smoothly as possible, changeover is not activated until the axis "is stationary", i.e. once the parameterized zero speed has been reached or undershot:

  DB31, ... DBX61.4 = 1 (axis/spindle stationary)

  MD36060 $MA_STANDSTILL_VELO_TOL (threshold velocity/speed 'axis/spindle stationary')

Parameter set changeover from the part program

For parameter set changeover from the part program, the user (machine manufacturer) must define corresponding user-specific auxiliary functions and evaluate them in the PLC user program. The PLC user program will then set the changeover request on the corresponding parameter set.

For detailed information on the auxiliary function output, see Section "H2: Auxiliary function outputs to PLC (Page 395)".

DB31, ... DBX9.3 (parameter set specification disabled by NC)

Parameter set changeover request will be ignored.

1.2.7 Signals from axis/spindle

DB31, ... DBX61.0 (drive test travel request)

If machine axes are traversed by special test functions such as "function generator", an explicit drive-test-specific enable is requested for the movement:

DB31, ... DBX61.0 == 1 (drive test travel request)

The motion is carried out once the motion is enabled:

DB31, ... DBX1.0 == 1 (drive test travel enable)

DB31, ... DBX61.3 (follow-up active)

The machine axis is in follow-up mode.

DB31, ... DBX61.4 (axis/spindle stationary ($n < n_{min}$)

"Axis/spindle stationary" is set by the NC if:

- No new setpoints are to be output **AND**
- The actual speed of the machine axis is lower than the parameterized zero speed:

  MD36060 $MA_STANDSTILL_VELO_TOL (threshold velocity axis stationary)

DB31, ... DBX61.5 (position controller active)

The machine axis position control loop is closed and position control is active.

DB31, ... DBX61.6 (speed controller active)

The machine axis speed control loop is closed and speed control is active.

DB31, ... DBX61.7 (current controller active)

The machine axis current control loop is closed and current control is active.

DB31, ... DBX69.0 / 69.1 / 69.2 (parameter set servo)

Active parameter set Coding accordingly:

DB31, ... DBX9.0 / 9.1 / 9.2 (controller parameter set selection)

DB31, ... DBX76.0 (lubrication pulse)

Following a control POWER ON/RESET, the signal status is 0 (FALSE).

The "lubrication pulse" is **inverted** (edge change), as soon as the machine axis has covered the parameterized traversing distance for lubrication:

MD33050 $MA_LUBRICATION_DIST (distance for lubrication by PLC)

1.2.8 Signals to axis/spindle (digital drives)

DB31, ... DBX21.0 / 21.1 / 21.2 (parameter set selection A, B, C)

Request to change over drive parameter set:

DBX 21.2	DBX 21.1	DBX21.0	Parameter set number
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

The feedback signal is sent via the interface signals:

DB31, ... DBX93.0,1 / 93.2 (active drive parameter set)

DB31, ... DBX21.3 / 21.4 (Motor selection A, B)

Selection of motor/operating mode.

DBX 21.4	DBX 21.3	Motor number	Mode
0	0	1	1
0	1	2	2
1	0	3	3
1	1	4	4

Only operating modes 1 and 2 are valid on main spindle drive:

- Operating modes 1: Star
- Operating modes 2: Delta

DB31, ... DBX21.5 (Motor selection done)

The PLC user program sends this signal to the drive to indicate successful motor selection. The pulses are then enabled by the drive.

DB31, ... DBX21.6 (integrator disable, n-controller)

The PLC user program inhibits the integrator of the speed controller for the drive. The speed controller is thus switched from PI to P controller.

Note

If the speed controller integrator disable is activated, compensations might take place in certain applications (e.g. if the integrator was already holding a load while stationary).

Feedback via the interface signal:

DB31, ... DBX93.6 = 1 (integrator n-controller disabled)

DB31, ... DBX21.7 (pulse enable)

The pulse enable for the drive module is only requested if all enable signals (hardware and software) are pending:

- Trigger equipment enable
- Controller and pulse enable
- Pulse enable (safe operating stop)
- Stored hardware input
- Setpoint enable
- "Ready to run state"
  - No drive alarm (DClink1 error)
  - DC link connected
  - Ramp-up completed

See also:
DB31, ... DBX93.7 (pulses enabled)

1.2.9 Signals from axis/spindle (digital drives)

DB31, ... DBX92.1 (ramp-function generator disable active)

The drive signals back to the PLC that ramp-function-generator fast stop is active. The drive is thus brought to a standstill without the ramp function (with speed setpoint 0).

DB31, ... DBX93.0, 1, 2 (active drive parameter set A, B, C)

The drive module sends this checkback signal to the PLC to indicate which drive parameter set is currently active. With bit combination A, B, C, eight different drive parameter sets can be selected by the PLC.

DB31, ... DBX93.3, 4 (active motor A, B)

The drive module (MSD) sends this checkback to the PLC to indicate which of the four motor types or motor operating modes is active.

The following selections can be made on the main spindle drive:

- Star mode (A=0, B=0)
- Delta mode (A=1, B=0)

DB31, ... DBX93.5 (DRIVE ready)

Checkback signal indicating that the drive is ready. The conditions required for traversing the axis/spindle are fulfilled.

DB31, ... DBX93.6 (integrator n-controller disabled)

The speed-controller integrator is disabled. The speed controller has thus been switched from PI to P controller.

DB31, ... DBX93.7 (pulses enabled)

The pulse enable for the drive module is available. The axis/spindle can now be traversed.

DB31, ... DBX94.0 (motor temperature prewarning)

The temperature of the motor is higher than the set motor temperature warning threshold (drive parameter p0604).

See also note below for "DB31, ... DBX94.1 (heat sink temperature prewarning)".

DB31, ... DBX94.1 (heat sink temperature prewarning)

The temperature of the heat sink in the power unit is outside the permissible range. If the overtemperature remains, the drive switches itself off after approx. 20 s.

Note

Temperature prewarning DB31, ... DBX94.0 and DBX94.1

The interface signals are derived from the following signals of the cyclic drive telegram:

- Case 1: Temperature warning in the message word
  - DB31, ... DBX94.0 = MELDW, bit 6 (no motor overtemperature warning)
  - DB31, ... DBX94.1 = MELDW, bit 7 (no thermal overload in power unit warning)
- Case 2: Warning of warning class B (only in interface mode "SIMODRIVE 611U", p2038 = 1)
  DB31, ... DBX94.0 == 1 and DBX94.1 == 1, if the following applies:
  Cyclic drive telegram, status word 1 (ZSW1), bits 11/12 == 2 (warning class B)

The interface signals are derived from the warning of warning class B if there is no specific information from the message word.

An alarm is displayed. Alarm number = 200.000 + alarm value (r2124)

For a detailed description of the motor temperature monitoring setting, see:

References:

- S120 Commissioning Manual, Section "Commissioning" > "Temperature sensors for SINAMICS components"
- S120 Function Manual, Section "Monitoring and protective functions"
- S120 List Manual
  - MELDW, bit 6 ≙ BO: r2135.14 → function diagram: 2548, 8016
  - MELDW, bit 7 ≙ BO: r2135.15 → function diagram: 2548, 2452, 2456, 8016

DB31, ... DBX94.2 (run-up completed)

The actual speed value is within the parameterized tolerance band again after changing the speed setpoint. The run-up procedure is now completed.

Any subsequent speed fluctuations, also outside the tolerance band, e.g. due to load changes, will not affect the interface signal.

DB31, ... DBX94.3 ($|M_d| < M_{dx}$)

The absolute value of the current torque $|M_d|$ is less than the parameterized threshold torque M_{dx} (torque threshold value 2, p2194).

The threshold torque is set as a percentage [%] of the current speed-dependent torque limitation.

DB31, ... DBX94.4 ($|n_{act}| < n_{min}$)

The actual speed value n_{act} is less than n_{min} (speed threshold value 3, p2161).

DB31, ... DBX94.5 ($|n_{act}| < n_x$)

The actual speed value n_{act} is less than n_x (speed threshold value 2, p2155).

DB31, ... DBX94.6 ($n_{act} = n_{set}$)

The actual speed value is within the tolerance band (p2163) surrounding the speed setpoint.

DB31, ... DBX95.7 (warning of warning class C is pending)

The drive signals that a warning of warning class C is pending.

1.3 Functions

1.3.1 Screen settings

Contrast, monitor type, foreground language, and display resolution to take effect after system startup can be set in the operator panel front machine data.

Contrast

MD9000 $MM_LCD_CONTRAST (contrast)

For slimline operator panel fronts with a **monochrome** LCD, the contrast to be applied following system startup can be set.

There are 16 different contrast settings (0: dark, 15: light).

Monitor type

MD9001 $MM_DISPLAY_TYPE (monitor type)

Indicate the relevant monitor type for optimum color matching.

Foreground language

MD9003 $MM_FIRST_LANGUAGE (foreground language)

In the case of SINUMERIK 840D sl, 2 languages are available simultaneously. The foreground language can be used to set the language to be displayed following control ramp-up.

The language can be changed in the DIAGNOSTICS operating area on the HMI user interface. Once the control has ramped up, the foreground language will be restored.

Display resolution

MD9004 $MM_DISPLAY_RESOLUTION (display resolution)

The number of places after the decimal point for the position display of the axes is defined in the display resolution. The position display consists of max. 12 characters including sign and decimal point. The number of digits after the decimal point can be set to between 0 and 5.

The default setting for the number of digits after the decimal point is 3, corresponding to a display resolution of 10–3 [mm] or [degrees].

REFRESH suppression

MD10131 $MN_SUPPRESS_SCREEN_REFRESH (screen refresh in case of overload)

Default setting for screen-refresh strategy with high NC utilization:

- Value 0: Refresh of current values is suppressed **in all channels**.
- Value 1: Refresh of current values is suppressed **in time-critical channels**.
- Value 2: Refresh of current values is **never** suppressed.

1.3.2 Settings for involute interpolation

Introduction

The involute of the circle is a curve traced out from the end point on a "piece of string" unwinding from the curve. Involute interpolation allows trajectories along an involute.

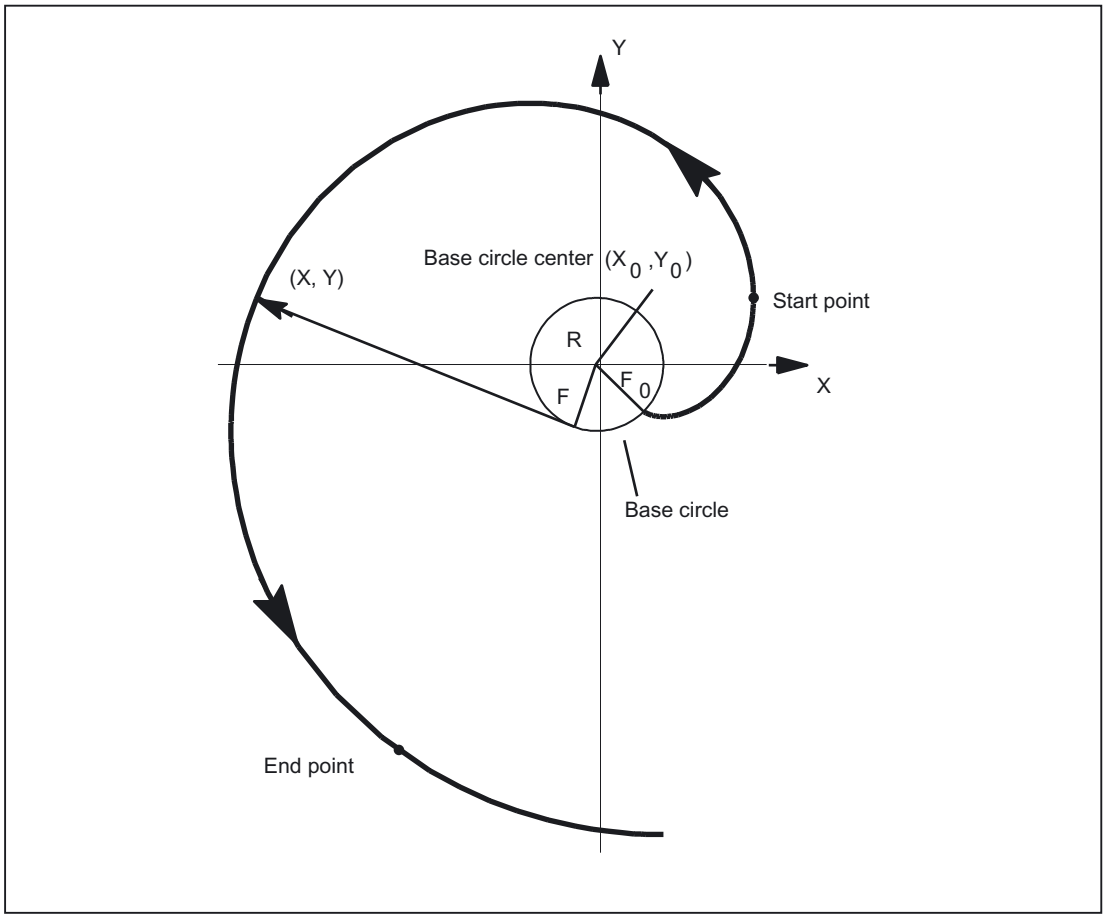


Figure 1-6 Involute (unwound from base circle)

Programming

A general description of how to program involute interpolation can be found in:

References:
Programming Manual, Fundamentals

In addition to the programmed parameters, machine data is relevant in two instances of involute interpolation; this data may need to be set by the machine manufacturer / end user.

Accuracy

If the programmed end point does not lie exactly on the involute defined by the starting point, interpolation takes place between the two involutes defined by the starting and end points (see illustration below).

The maximum deviation of the end point is determined by the machine data:

MD21015 $MC_INVOLUTE_RADIUS_DELTA (end point monitoring for involute)

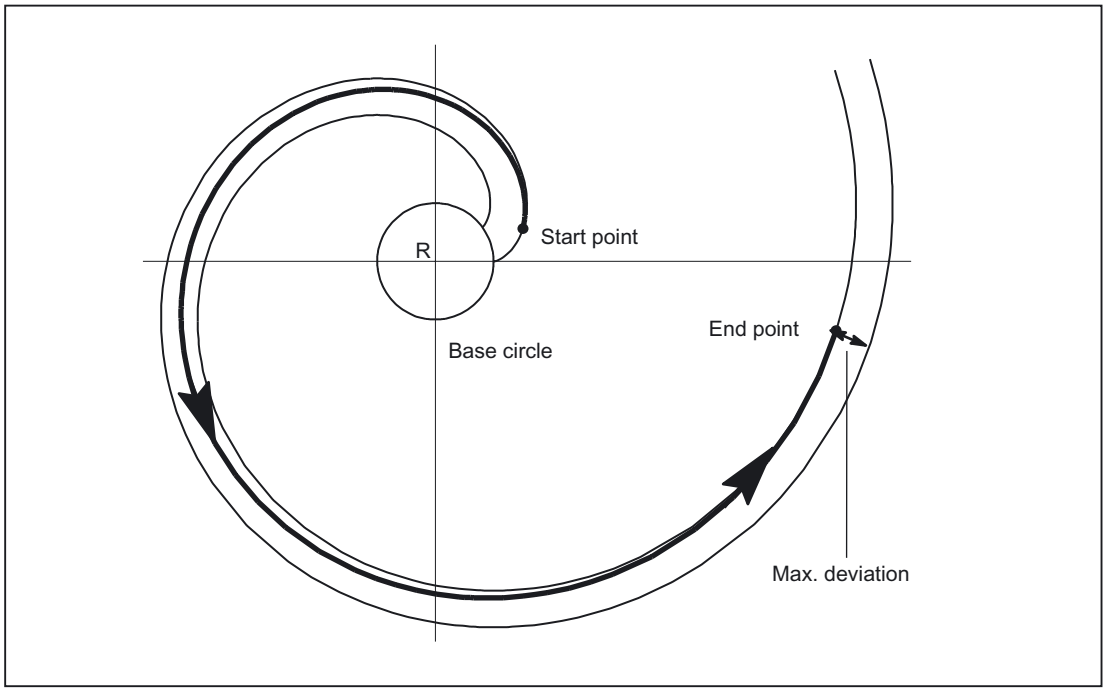


Figure 1-7 MD21015 specifies the max. permissible deviation

Limit angle

If AR is used to program an involute leading to the base circle with an angle of rotation that is greater than the maximum possible value, an alarm is output and program execution aborted.

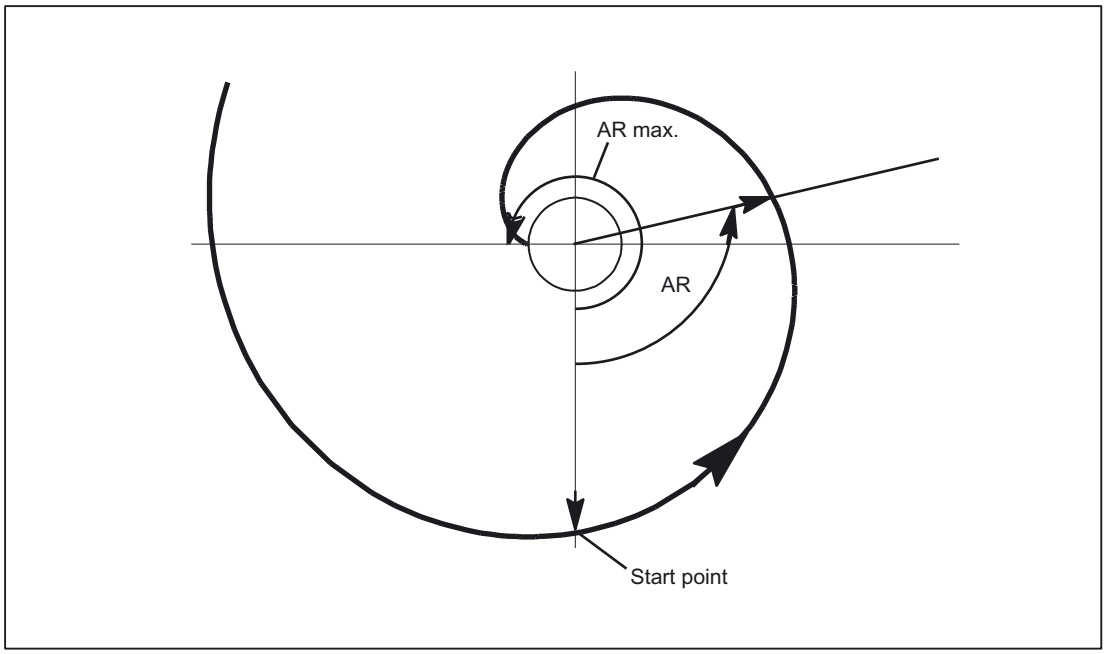


Figure 1-8 Limited angle of rotation towards base circle

The alarm display can be suppressed using the following parameter settings:

MD21016 $MC_INVOLUTE_AUTO_ANGLE_LIMIT = TRUE (automatic angle limitation for involute interpolation)

The programmed angle of rotation is then also limited automatically and the interpolated path ends at the point at which the involute meets the base circle. This, for example, makes it easier to program an involute which starts at a point outside the base circle and ends directly on it.

Tool radius compensation

2 1/2 D tool radius compensation is the only tool radius compensation function permitted for involutes. If 3D tool radius compensation is active (both circumferential and face milling), when an involute is programmed, machining is interrupted with alarm 10782.

With 2 1/2 D tool radius compensation, the plane of the involute must lie in the compensation plane. or else alarm 10781 will be generated. It is however permissible to program an additional helical component for an involute in the compensation plane.

Dynamic response

Involutes that begin or end on the base circle have an infinite curvature at this point. To ensure that the velocity is adequately limited at this point when tool radius compensation is active, without reducing it too far at other points, the "Velocity limitation profile" function must be activated:

MD28530 $MC_MM_PATH_VELO_SEGMENTS > 1 (number of memory elements for limiting the path velocity)

A setting of 5 is recommended. This setting need not be made if only involute sections are used which have radii of curvature that change over a relatively small area.

1.3.3 Activate DEFAULT memory

GUD start values

The DEF... / REDEF... NC commands can be used to assign default settings to global user data (GUD). These default settings must be permanently stored in the system if they are to be available after certain system states (e.g. RESET).

The memory space for this is taken from the memory area that was assigned via the machine data:

MD18150 $MM_GUD_VALUES_MEM (number of additional parameters according to MD18170)

The setting for activating the stored default values is made in machine data:

MD11270 $MN_DEFAULT_VALUES_MEM_MASK (activation default values for NC language elements)

References:

Function Manual, Extended Functions; S7: "Memory Configuration"

Programming Manual, Job Planning

1.3.4 Read/write PLC variable

High-speed data channel

For high-speed exchange of information between the PLC and NC, a memory area is reserved in the communications buffer on these modules (dual-port RAM). Variables of any type (I/O, DB, DW, flags) may be exchanged within this memory area.

The PLC accesses this memory using 'Function Calls' (FC) while the NC uses system variables.

Organization of memory area

The user's programming engineer (NC and PLC) is responsible for organizing (structuring) this memory area.

Every storage position in the memory can be addressed provided that the limit is selected according to the appropriate data format (i.e. a DWORD for a 4-byte limit, a WORD for a 2-byte limit, etc.).

The memory is accessed via the data type and the position offset within the memory area.

Access from NC

System variables are available in the NC for fast access to PLC variables from a part program or synchronized action. The data is read/written directly by the NC. The data type results from the identifier of the system variables. The position within the memory area is specified as index in bytes.

System variable	Data type	Range of values
$A_DBB[<index>]	Byte (8 bits)	0 <= x <= 255
$A_DBW[<index>]	Word (16 bits)	-32768 <= x <= 32767
$A_DBD[<index>]	Double word (32 bits)	-2147483648 <= x <= 2147483647
$A_DBR[<index>]	Floating point (32 bits)	$\pm(1.5 \cdot 10^{-45} <= x <= 3.4 \cdot 10^{38})$

Access from PLC

The PLC uses function calls (FC) to access the memory. The data is read and written immediately in the DPR with the FC and not just at the beginning of the PLC cycle. Data type and position in the memory area are transferred as parameters to the FC.

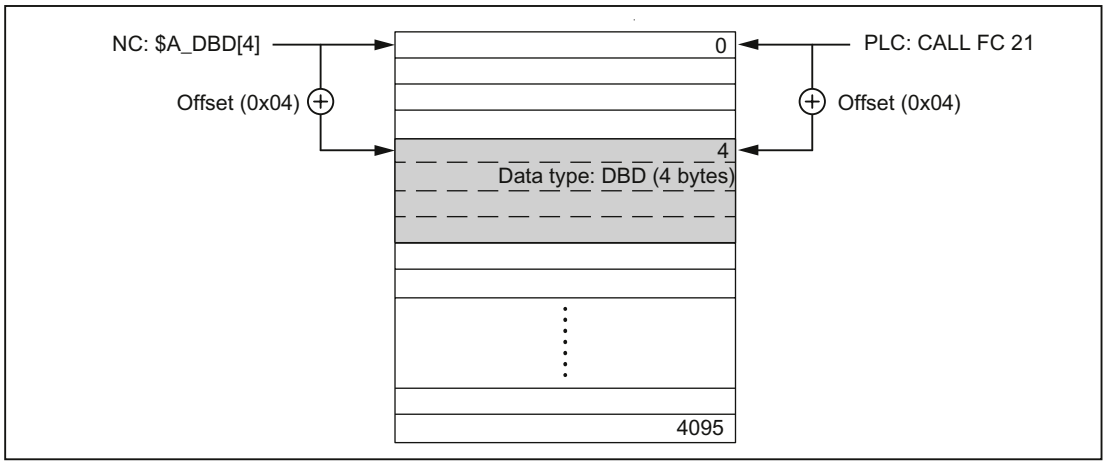


Figure 1-9 Communications buffer (DPR) for NC/PLC communication

Supplementary conditions

- The structuring of the DPR memory area is the sole responsibility of the user. No checks are made for matching configuration.
- A total of 4096 bytes are available in the input and output directions.
- Single-bit operations are not supported and must be linked back to byte operations by the user.
- Since the contents of variables are manipulated directly in the communications buffer, the user must remember that intermediate changes in values occur as a result of multiple access operations where a variable is evaluated several times or when variables are linked (i.e. it may be necessary to store values temporarily in local variables or R parameters or to set up a semaphore).
- The user's programming engineer is responsible for coordinating access operations to the communications buffer from different channels.
- Data consistency can be guaranteed only for access operations up to 16 bits (byte and word). The user is solely responsible for ensuring consistent transmission of 32-bit variables (double and real). A simple semaphore mechanism is available in the PLC for this purpose.

- The PLC stores data in 'Little Endian' format in the DPR.
- Values transferred with $A_DBR are subject to data conversion and hence to loss of accuracy. The data format for floating-point numbers is DOUBLE (64 bits) in the NC, but only FLOAT (32 bits) in the PLC. The format used for storage in the dual-port RAM is FLOAT. Conversion takes place respectively before/after storage in the dual-port RAM.

  If a read/write access is made from the NC to a variable in the dual-port RAM, the conversion is performed twice. It is impossible to prevent differences between read and written values because the data is stored in both formats.

  Example

  Bypassing the problem by means of comparison on "EPSILON" (minor deviation)

Program code

```
N10          DEF REAL DBR
N12          DEF REAL EPSILON = 0.00001
N20          $A_DBR[0]=145.145
N30          G4 F2
N40          STOPRE
N50          DBR=$A_DBR[0]
N60          IF ( ABS(DBR/145.145-1.0) < EPSILON ) GOTOF ENDE
N70          MSG ( "error" )
N80          M0
N90          END:
N99          M30
```

Activation

The maximum number of simultaneously writable output variables is adjustable via:
MD28150 $MC_MM_NUM_VDIVAR_ELEMENTS (number of elements for writing PLC variables)

Example

A variable of type WORD is to be transferred from the PLC to the NC.

The position offset within the NC input (PLC output area) should be the fourth byte. The position offset must be a whole-number multiple of the data width.

Writing from PLC:

```
Program code                                    Comment
. . .
CALL FC21 (
Enable :=M10.0,                                 ; if TRUE, then FC21 active
Funct :=B#16#4,
S7Var :=P#M 104.0 WORD1,
IVAR1 :=04,
IVAR2 :=-1,
Error :=M10.1,
ErrCode :=MW12);
. . .
)
```

Reading in part program

```
Program code                                    Comment
. . .
PLCDATA = $A_DBW[4];                            ; Read a word
. . .
```

Behavior during POWER ON, block search

The DPR communications buffer is initialized during "POWER ON".

During a "block search", the PLC variable outputs are collected and transferred to the DPR communications buffer with the approach block (analogous to writing of analog and digital outputs).

Other status transitions have no effect in this respect.

References

A detailed description of the data exchange by the PLC with FC 21 can be found in:

SINUMERIK 840D sl: Section "FC 21: transfer PLC NCK data exchange (Page 1138)"

1.3.5 Access protection via password and keyswitch

1.3.5.1 Access protection via password and keyswitch

Access authorization

Access to functions, programs and data is useroriented and controlled via 8 hierarchical protection levels. These are subdivided into:

- Password levels for Siemens, machine manufacturer and end user
- Keyswitch positions for end user

Multi-level security concept

A multi-level security concept to regulate access rights is available in the form of password levels and keyswitch settings.

Protection level	Type	User	Access to (examples)
0	Password	Siemens	All functions, programs and data
1	Password	Machine manufacturer: Development	defined functions, programs and data; for example: entering options
2	Password	Machine manufacturer: Startup engineer	defined functions, programs and data; for example: Bulk of machine data
3	Password	End user: Service	Assigned functions, programs and data
4	Keyswitch position 3	End user: Programmer, machine setter	less than the protection level 0 to 3; established by the machine manufacturer or end user
5	Keyswitch position 2	End user: Skilled operator without programming knowledge	less than the protection level 0 to 3; established by the end user
6	Keyswitch position 1	End user: Trained operator without programming knowledge	Example: Program selection only, tool wear entry, and work offset entry
7	Keyswitch position 0	End user: Semi-skilled operator	Example: no inputs and program selection possible, only machine control panel operable

Access features

- Protection level 0 provides the greatest number of access rights, protection level 7 the least.
- If certain access rights are granted to a protection level, these protection rights automatically apply to any higher protection levels.
- Conversely, protection rights for a certain protection level can only be altered from a higher protection level.
- Access rights for protection levels 0 to 3 are permanently assigned by Siemens and cannot be altered (default).
- Access rights can be set by querying the current keyswitch positions and comparing the passwords entered. When a password is entered it overwrites the access rights of the keyswitch position.
- Options can be protected on each protection level. However, option data can only be entered in protection levels 0 and 1.
- Access rights for protection levels 4 to 7 are only suggestions and can be altered by the machine tool manufacturer or end user.

1.3.5.2 Password

Set password

The password for a protection level (0 – 3) is entered via the HMI user interface.

Example:

DIAGNOSTIC operating area, softkey: SET PASSWORD

References:

Commissioning Manual SINUMERIK 840D sl base software and HMI sl

Delete password

Access rights assigned by means of setting a password remain effective until they are explicitly revoked by deleting the password.

Example:

DIAGNOSTIC operating area, softkey: DELETE PASSWORD

References:

Commissioning Manual SINUMERIK 840D sl base software and HMI sl

Note

Access rights and password status (set/deleted) are not affected by POWER OFF/ON!

Maximum number of characters

A password may contain up to eight characters. We recommend that you confine yourself to the characters available on the operator panel front when defining the password. Where a password consists of less than eight characters, the additional characters are interpreted as blanks.

Defaults

The following default passwords are defined for protection levels 1 to 3:

- Protection level 1: SUNRISE
- Protection level 2: EVENING
- Protection level 3: CUSTOMER

Note

Following NC-CPU ramp-up in commissioning mode (NCK commissioning switch: position 1) the passwords for protection levels 1 – 3 are reset to the default settings. For reasons of data protection, we strongly recommend that you change the default settings.

1.3.5.3 Keyswitch positions (DB10, DBX56.4 to 7)

Key switch

The keyswitch has four positions, to which protection levels 4 to 7 are assigned. The keyswitch comprises a number of keys in a variety of colors which can be set to different switch positions.

Switch position	Retraction pos.	DB10, DBB56	Protection level
Position 0 	-	Bit 4	7
Position 1 	0 or 1 black key	Bit 5	6
Position 2 	0 or 1 or 2 green key	Bit 6	5
Position 3 	0 or 1 or 2 or 3 red key	Bit 7	4

Figure 1-10 Switch positions 0 to 3

Switch positions

Switch position 0 has the most restricted access rights. Switch position 3 has the least restricted access rights.

DB10, DBX56.4 / .5 / .6 / .7 (switch positions 0 / 1 / 2 / 3)

Machine-specific enables for access to programs, data and functions can be assigned to the switch positions. For detailed information, please refer to:

References

- CNC Commissioning Manual: NCK, PLC, Drives, Fundamentals,
  Section: Basics on the protection levels
- Commissioning Manual SINUMERIK Operate (IM9); General Settings,
  Section: Access levels

Default settings via the PLC user program

The keyswitch positions are transferred to the NC/PLC interface via the basic PLC program. The corresponding interface signals can be modified via the PLC user program. In this context, from the point of view of the NC, only one switch position should ever be active, i.e. the corresponding interface signal set to 1. If, from the point of view of the NC, a number of switch positions are active at the same time, switch position 3, i.e. the keyswitch position with the least restricted access rights, will be activated internally by the NC.

1.3.5.4 Parameterizable protection levels

Parameterizable protection level

The parameter level can be freely parameterized for a variety of functions and data areas. The protection level is set via operator-panel machine data, designated as follows:
$MM_USER_CLASS_<*Function_DataArea*>

Examples:

$MM_USER_CLASS_READ_TOA	Read tool offsets
$MM_USER_CLASS_WRITE_TOA	Write tool offsets
$MM_USER_CLASS_READ_PROGRAM	Read part programs
$MM_USER_CLASS_WRITE_PROGRAM	Write/edit part programs

Default values

On delivery or following standard commissioning, with very few exceptions, the default value for the protection level will be set to 7, i.e. the lowest protection level.

1.3.6 "Parking" of a machine axis

In the "Parking" state, a machine axis can be moved mechanically or maintenance performed (e.g. encoder replacement), without triggering an alarm. For this purpose, the axis-specific NC/PLC interface signals for the active position measuring system and the controller enable of the machine axis must be reset:

- DB31, ... DBX1.5/.6 = 0 (position measuring system 1/2)
- DB31, ... DBX2.1 = 0 (controller enable)

The encoder status of the active measuring system of the axis is then displayed as "Not referenced":

- DB31, ... DBX60.4/.5 == 0 (referenced/synchronized, encoder 1/2)

Canceling "Parking"

The "Parking" state is canceled by setting the NC/PLC interface signals:

- DB31, ... DBX1.5/.6 = 1 (position measuring system 1/2)
- DB31, ... DBX2.1 = 1 (controller enable)

The position control of the machine axis then becomes active again at the current position.

The encoder state of the measuring system depends on the measuring system type:

- Incremental measuring system: "Not referenced" state

  DB31, ... DBX60.4/.5 == 0 (referenced/synchronized, encoder 1/2)
- Absolute measuring system: "Referenced" state
  DB31, ... DBX60.4/.5 == 1 (referenced/synchronized, encoder 1/2)

Incremental measuring systems: Encoder state "Referenced"

To reach the encoder state "Referenced", incremental measuring systems must be re-referenced.

⚠ WARNING

If changes have been made on the measuring system during parking that require a change of the parameterized machine data, e.g. mounting of another encoder, the measuring system must be completely remeasured and referenced. See Section "R1: Referencing (Page 1319)".

1.3.7 Switchover of motor/drive data sets

1.3.7.1 General Information

Behavior as of SW 4.5 SP1

Up to and including SW 4.5, the format of the interfaces for the motor and drive data set switchover in the NC/PLC interface was fixed.

As of SW 4.5 SP1, the quantity structure of the programmable motor data sets and drive data sets per motor data set and therefore the rigid assignment of the signals to the motor and drive data set switchover in the NC/PLC interface signals has become more flexible.

Motor and drive data sets

For optimum adaptation to the particular machining situation or because of different machine configurations, it may be necessary that several different data sets are available in a drive for motors, drive parameters and encoders. The creation of the basic data sets of the drive objects is performed during startup with the aid of the "Drive wizard".

Note

References

Commissioning Manual: CNC: NCK, PLC, Drive, Section "Commissioning NC-controlled drives"

The following duplication and management of the data sets is performed via the user interface:

SINUMERIK Operate: Operating area "Start-up" > "Drive system" > "Drives" > "Data sets"

The activation of the motor data set (MDS) or drive data set (DDS) required for a machine axis in a specific machining situation, must be made from the PLC user program via the interfaces described below.

Axial NC/PLC interface

The interfaces in the axial NC/PLC interface for switching the motor and drive data sets is divided into three areas:

- Validity and format of the request/display interfaces (Page 71)
- Request for a new motor data set and/or drive data set (Page 72)
- Display of the active motor and/or drive data set (Page 72)

1.3.7.2 Validity and format of the request/display interfaces

Validity

As soon as the control has received all the required information from the drive and has been evaluated by the NC, the request and display interfaces are shown as valid:

DB31, ... DBX130.7 == 1 (interface is valid)

If no or incompatible information is transferred from the drive, the request and display interfaces remain invalid.

Note

With invalid request and display interfaces, it is the sole responsibility of the user / machine manufacturer to perform a data set switchover via the request and display interfaces.

Format

The current format of the interfaces, i.e. which of the five bits of the request and display interfaces are used for the addressing of the motor data set and which for the drive data set, depends on the number of motor and drive data sets in the drive. The format is specified via:

DB31, ... DBX130.0-4, with bit x = <value>

<value>	Meaning
0	Bit position for motor data set switchover (MDS) or invalid bit position
1	Bit position for drive data set switchover (DDS)

See also

Example (Page 73)

Overview of the interfaces (Page 74)

1.3.7.3 Request for a new motor data set and/or drive data set

The request to activate a specific motor and drive data set is performed via:

DB31, ... DBX21.0 - .4 = <MDS/DDS index>

Range of values

The addressing of a data set is performed via the index i, with i = 0, 1, 2, ...

Data set	Range of values, index i
MDS[i]	0 ≤ i ≤ 32
DDS[i]	0 ≤ i ≤ 32

Formatting

The formatting of the interface, i.e. which bits are available for the motor data set index and which for the drive data set index is displayed via:

DB31, ... DBX130.0 - .4 (see Section "Validity and format of the request/display interfaces (Page 71)")

Specific number of data sets

The specific number of motor and drive data sets available in the motor can be determined via the following drive parameters:

- p0130 (number of motor data sets)
- p0180 (number drive data sets)

1.3.7.4 Display of the active motor and/or drive data set

Display of the active motor and/or drive data set

The index of an active data set can be read via:

DB31, ... DBX93.0 - .4

Value range and formatting are identical to the request interface. See Section "Request for a new motor data set and/or drive data set (Page 72)"

1.3.7.5 Example

Two motor data sets (MDS) and two drive data sets (DDS) are available in the drive. This corresponds to "No.": 9 of the possible data set combinations displayed in Figure 1-11 Motor/drive data set switchover (Page 74).

Format

Bit positions for drive data set switchover (DDS):

- DB31, ... DBX130.0 == 1

Bit positions for motor data set switchover (MDS):

- DB31, ... DBX130.1 == 0

Invalid bit positions:

- DB31, ... DBX130.2 == 0
- DB31, ... DBX130.3 == 0
- DB31, ... DBX130.4 == 0

Interfaces of the drive data sets (DDS)

Relevant bit positions of the request and display interfaces:

- DB31, ... DBX21.0 / DBX93.0
  - DB31, ... DBX21.0 / DBX93.0 == 0 ⇒ 1st drive data set DDS[0]
  - DB31, ... DBX21.0 / DBX93.0 == 1 ⇒ 2nd drive data set DDS[1])

Interfaces of the motor data sets (MDS)

Relevant bit positions of the request and display interfaces:

- DB31, ... DBX21.1 / DBX93.1
  - DB31, ... DBX21.1 / DBX93.1 == 0 ⇒ 1st motor data set MDS[0]
  - DB31, ... DBX21.1 / DBX93.1 == 1 ⇒ 2nd motor data set MDS[1])

Invalid bit positions (MDS/DDS)

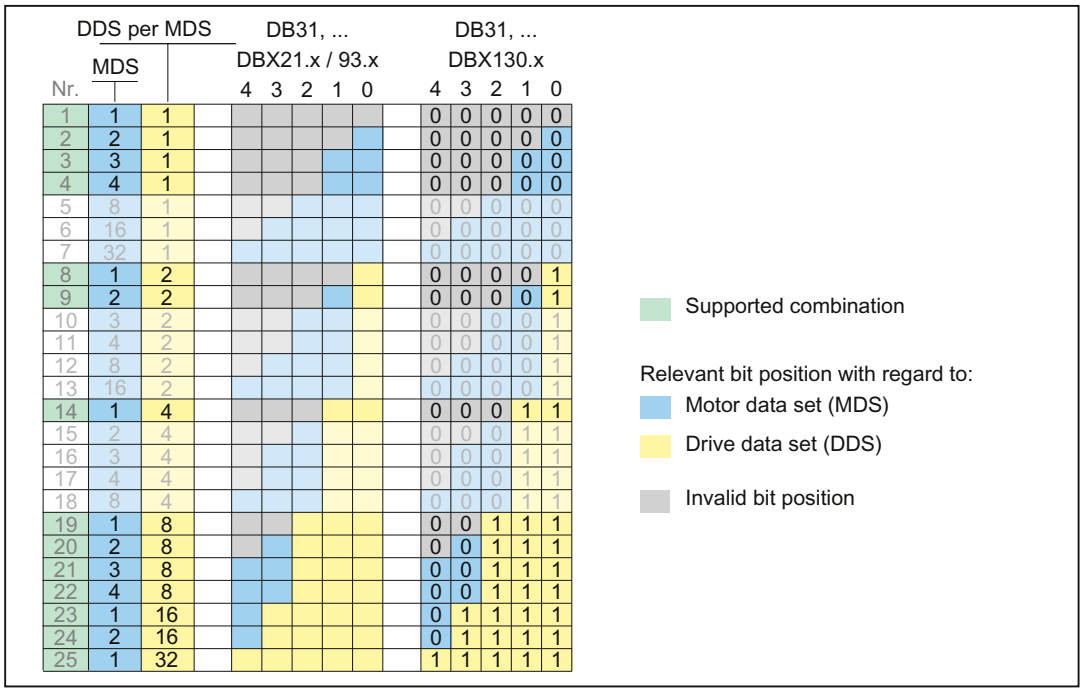
Invalid bit positions of the request and display interfaces:

- DB31, ... DBX21.1 / DBX93.2
- DB31, ... DBX21.1 / DBX93.3
- DB31, ... DBX21.1 / DBX93.4

See also

Overview of the interfaces (Page 74)

1.3.7.6 Overview of the interfaces



Nr.	MDS	DDS per MDS	DB31, ... DBX21.x / 93.x (4 3 2 1 0)	DB31, ... DBX130.x (4 3 2 1 0)
1	1	1		0 0 0 0 0
2	2	1		0 0 0 0 0
3	3	1		0 0 0 0 0
4	4	1		0 0 0 0 0
5	8	1		0 0 0 0 0
6	16	1		0 0 0 0 0
7	32	1		0 0 0 0 0
8	1	2		0 0 0 0 1
9	2	2		0 0 0 0 1
10	3	2		0 0 0 0 1
11	4	2		0 0 0 0 1
12	8	2		0 0 0 0 1
13	16	2		0 0 0 0 1
14	1	4		0 0 0 1 1
15	2	4		0 0 0 1 1
16	3	4		0 0 0 1 1
17	4	4		0 0 0 1 1
18	8	4		0 0 0 1 1
19	1	8		0 0 1 1 1
20	2	8		0 0 1 1 1
21	3	8		0 0 1 1 1
22	4	8		0 0 1 1 1
23	1	16		0 1 1 1 1
24	2	16		0 1 1 1 1
25	1	32		1 1 1 1 1

MDS	Number of motor data sets
DDS per MDS	Number of drive data sets per motor data set
DB31, ... DBX21.x	Request interface of the data sets to be activated
DB31, ... DBX93.x	Display interface of the active data sets
DB31, ... DBX130.x	Interface for displaying the formatting of the request and display interfaces

Figure 1-11 Motor/drive data set switchover

1.3.7.7 Supplementary conditions

Variable number of drive data sets for the "last" motor data set

The "last" motor data set is the motor data set with the highest number or index.

Generally, the same number of drive data sets is created for each motor data set (number of "DDS per MDS") in the drive. Only the "last" motor data set can differ from this; any number (a) of drive data sets can be parameterized:

1 ≤ a ≤ (number of "DDS per MDS")

Example

Four motor data sets (MDS) and eight drive data sets (DDS) per motor data set (DDS per MDS) are to be parameterized. This corresponds to "No.": 22 of the possible data set combinations displayed in Figure 1-11 Motor/drive data set switchover (Page 74):

- Motor data sets: MDS[0], MDS[1], ... MDS[3] ("last" motor data set)
- Drive data sets per motor data set: DDS[0] ... DDS[7]

The number of drive data sets for the individual motor data sets is therefore:

Motor data set	Number of drive data sets per motor data set
MDS[0] ... MDS[2]	8 DDS
MDS[3]	1 - 8 DDS

See also

Overview of the interfaces (Page 74)

1.4 Examples

Parameter set changeover

A parameter-set changeover is performed to change the position-control gain (servo gain factor) for machine axis X1 from v = 4.0 to Kv = 0.5.

Preconditions

The parameter set changeover must be enabled by the machine data:

MD35590 $MA_PARAMSET_CHANGE_ENABLE [AX1] = 1 or 2 (parameter set change possible)

The 1st parameter set for machine axis X1 is set, in accordance with machine data with index "0" NC/PLC interface:

DB31, … DBX9.0 - DBX9.2 = 0 (controller parameter set)

Parameter-set-dependent machine data

Parameter-set-dependent machine data are set as follows:

Machine data	Comment
MD32200 $MA_POSCTRL_GAIN [0, AX1] = 4.0	Servo gain setting for parameter set 1
MD32200 $MA_POSCTRL_GAIN [1, AX1] = 2.0	Servo gain setting for parameter set 2
MD32200 $MA_POSCTRL_GAIN [2, AX1] = 1.0	Servo gain setting for parameter set 3
MD32200 $MA_POSCTRL_GAIN [3, AX1] = 0.5	Servo gain setting for parameter set 4
MD32200 $MA_POSCTRL_GAIN [4, AX1] = 0.25	Servo gain setting for parameter set 5
MD32200 $MA_POSCTRL_GAIN [5, AX1] = 0.125	Servo gain setting for parameter set 6
MD31050 $MA_DRIVE_AX_RATIO_DENOM [0, AX1] = 3	Denominator load gearbox for parameter set 1
MD31050 $MA_DRIVE_AX_RATIO_DENOM [1, AX1] = 3	Denominator load gearbox for parameter set 2
MD31050 $MA_DRIVE_AX_RATIO_DENOM [2, AX1] = 3	Denominator load gearbox for parameter set 3
MD31050 $MA_DRIVE_AX_RATIO_DENOM [3, AX1] = 3	Denominator load gearbox for parameter set 4
MD31050 $MA_DRIVE_AX_RATIO_DENOM [4, AX1] = 3	Denominator load gearbox for parameter set 5
MD31050 $MA_DRIVE_AX_RATIO_DENOM [5, AX1] = 3	Denominator load gearbox for parameter set 6

Machine data	Comment
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [0, AX1] = 5	Counter load gearbox for parameter set 1
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [1, AX1] = 5	Counter load gearbox for parameter set 2
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [2, AX1] = 5	Counter load gearbox for parameter set 3
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [3, AX1] = 5	Counter load gearbox for parameter set 4
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [4, AX1] = 5	Counter load gearbox for parameter set 5
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [5, AX1] = 5	Counter load gearbox for parameter set 6
MD35130 $MA_AX_VELO_LIMIT [0...5, AX1]	Setting for each parameter set*)
MD32800 $MA_EQUIV_CURRCTRL_TIME [0..5, AX1]	Setting for each parameter set*)
MD32810 $MA_EQUIV_SPEEDCTRL_TIME [0..5, AX1]	Setting for each parameter set*)
MD32910 $MA_DYN_MATCH_TIME [0...5, AX1]	Setting for each parameter set*)
*) The appropriate line must be specified separately for each parameter set according to the applicable syntax rules.	

Changeover

In order to switch over the position-control gain, the PLC user program selects the 4th parameter set for machine axis X1.

- Request by PLC user program:

  DB31, … DBX9.0 – DBX9.2 = 3 (parameter set servo)

  – A request to change over to the 4th parameter set is sent for machine axis AX1.
  – The parameter set is changed over once a delay has elapsed.
  – Parameter set 4 is now active, in accordance with machine data with index "3"

- Feedback by NC:

  DB31, … DBX69.0 – DBX69.2 = 3 (parameter set servo)

  – The NC confirms/acknowledges the parameter-set changeover.

1.5 Data lists

1.5.1 Machine data

1.5.1.1 Display machine data

Number	Identifier: $MM_	Description
SINUMERIK Operate		
9000	LCD_CONTRAST	Contrast
9001	DISPLAY_TYPE	Monitor type
9004	DISPLAY_RESOLUTION	Display resolution

1.5.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10350	FASTIO_DIG_NUM_INPUTS	Number of active digital NCK input bytes
10360	FASTIO_DIG_NUM_OUTPUTS	Number of active digital NCK output bytes
10361	FASTIO_DIG_SHORT_CIRCUIT	Short-circuit digital inputs and outputs
11120	LUD_EXTENDED_SCOPE	Activate programglobal variables (PUD)
11270	DEFAULT_VALUES_MEM_MSK	Activ. Function: Save DEFAULT values of GUD.
18150	MM_GUD_VALUES_MEM	Reserve memory space for GUD

1.5.1.3 Channelspecific machine data

Number	Identifier: $MC_	Description
21015	INVOLUTE_RADIUS_DELTA	NC start disable without reference point
21016	INVOLUTE_AUTO_ANGLE_LIMIT	Automatic angle limitation for involute interpolation
27800	TECHNOLOGY_MODE	Technology in channel
28150	MM_NUM_VDIVAR_ELEMENTS	Number of write elements for PLC variables
28530	MM_PATH_VELO_SEGMENTS	Number of storage elements for limiting path velocity in block

1.5.1.4 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30350	SIMU_AX_VDI_OUTPUT	Output of axis signals for simulation axes
33050	LUBRICATION_DIST	Lubrication pulse distance
35590	PARAMSET_CHANGE_ENABLE	Parameter set definition possible from PLC
36060	STANDSTILL_VELO_TOL	Maximum velocity/speed when axis/spindle stationary
36610	AX_EMERGENCY_STOP_TIME	Length of the braking ramp for error states
36620	SERVO_DISABLE_DELAY_TIME	Cutout delay servo enable

1.5.2 System variables

Names	Description
$P_FUMB	Unassigned part program memory (**F**ree **U**ser **M**emory **B**uffer)
$A_DBB[n]	Data on PLC (data type BYTE)
$A_DBW[n]	Data on PLC (WORD type data)
$A_DBD[n]	Data on PLC (DWORD type data)
$A_DBR[n]	Data on PLC (REAL type data)

1.5.3 Signals

1.5.3.1 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Keyswitch setting 0 to 3	DB10.DBX56.4-7	DB2600.DBX0.4-7

1.5.3.2 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Remote diagnostics active (HMI alarm is pending)	DB10.DBX103.0	-
AT box ready	DB10.DBX103.5	-
HMI temperature limit	DB10.DBX103.6	-
HMI battery alarm	DB10.DBX103.7	-
NCK--Ready	DB10.DBX104.7	-
HMI2-CPU-Ready E_MMC2 Ready	DB10.DBX108.1	-
HMI CPU1 Ready (HMI to MPI)	DB10.DBX108.2	-
HMI1-CPU at OPI Ready	DB10.DBX108.3	DB2700.DBX2.3
Drives in cyclic operation	DB10.DBX108.5	DB2700.DBX2.5
Drives ready	DB10.DBX108.6	DB2700.DBX2.6
NC Ready	DB10.DBX108.7	DB2700.DBX2.7
NCK alarm is active	DB10.DBX109.0	DB2700.DBX3.0
NCU heat sink temperature alarm	DB10.DBX109.5	-
Air temperature alarm	DB10.DBX109.6	DB2700.DBX3.6
NCK battery alarm	DB10.DBX109.7	-

1.5.3.3 Signals to operator panel front

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Screen bright	DB19.DBX0.0	-
Screen dark	DB19.DBX0.1	-
Key disable	DB19.DBX0.2	DB1900.DBX5000.2
Delete Cancel alarms (HMI Advanced only)	DB19.DBX0.3	-
Delete Recall alarms (HMI Advanced only)	DB19.DBX0.4	-
Actual value in WCS	DB19.DBX0.7	DB1900.DBX5000.7
Unload part program	DB19.DBX13.5	-
Load part program	DB19.DBX13.6	-
Part program selection	DB19.DBX13.7	DB1700.DBX1000.7
File system active/passive	DB19.DBX14.7	-
Part program handling: Number of the control data	DB19.DBX16.7	DB1700.DBX1001.7 == 1
Mode change disable	DB19.DBX44.0	-

1.5.3.4 Signals from operator panel front

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Screen is dark	DB19.DBX20.1	-
Switch over MCS/WCS	DB19.DBX20.7	DB1900.DBX0.7
Error (Part program handling status)	DB19.DBX26.0	DB1700.DBX2000.2
OK (Part program handling status)	DB19.DBX26.1	DB1700.DBX2000.1
Active (Part program handling status)	DB19.DBX26.3	DB1700.DBX2000.3
Unload (Part program handling status)	DB19.DBX26.5	DB1700.DBX2000.5
Load (Part program handling status)	DB19.DBX26.6	DB1700.DBX2000.6
Select (Part program handling status)	DB19.DBX26.7	DB1700.DBX2000.7
FC9: Start measuring in Jog	DB19.DBX42.0	-
FC9 Out: Active	DB19.DBX45.0	-
FC9 Out: Done	DB19.DBX45.1	-
FC9 Out: Error	DB19.DBX45.2	-
FC9 Out: StartErr	DB19.DBX45.3	-

1.5.3.5 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Delete distancetogo (channelspecific)	DB21, ...DBX6.2	DB3200.DBX6.2

1.5.3.6 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Channelspecific NCK alarm is active	DB21, ...DBX36.6	DB3300.DBX4.6
NCK alarm with processing stop present	DB21, … .DBX36.7	DB3300.DBX4.7
Overstore active	DB21, ...DBX318.7	-

1.5.3.7 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Axis/spindle disable	DB31, ...DBX1.3	DB380x.DBX1.3
Follow-up mode	DB31, ...DBX1.4	DB380x.DBX1.4
Position measuring system 1	DB31, ...DBX1.5	DB380x.DBX1.5
Position measuring system 2	DB31, ...DBX1.6	DB380x.DBX1.6
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Delete distance-to-go (axis-specific) / spindle reset	DB31, ...DBX2.2	DB380x.DBX2.2
Motor/drive data set: Selection	DB31, ...DBX21.0-4	DB380x.DBX4001.0-4
Motor being selected	DB31, ...DBX21.5	-
Speed controller integrator disable	DB31, ...DBX21.6	DB380x.DBX4001.6
Pulse enable	DB31, ...DBX21.7	DB380x.DBX4001.7

1.5.3.8 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized, encoder 1/2	DB31, ...DBX60.4/5	DB390x.DBX0.4/5
Traversing command minus/plus	DB31, ...DBX64.6/7	DB390x.DBX4.6/7
Follow up active	DB31, ...DBX61.3	DB390x.DBX1.3
Axis/spindle stationary ($n < n_{min}$)	DB31, ...DBX61.4	DB390x.DBX1.4
Position controller active	DB31, ...DBX61.5	DB390x.DBX1.5
Speed controller active	DB31, ...DBX61.6	DB390x.DBX1.6
Current controller active	DB31, ...DBX61.7	DB390x.DBX1.7
Lubrication pulse	DB31, ...DBX76.0	DB390x.DBX1002.0
Active motor/drive data set	DB31, ...DBX93.0-4	DB390x.DBX4001.0-4
Drive ready	DB31, ...DBX93.5	DB390x.DBX4001.5
Speed controller integrator disabled	DB31, ...DBX93.6	DB390x.DBX4001.6
Pulses enabled	DB31, ...DBX93.7	DB390x.DBX4001.7
Motor temperature prewarning	DB31, ...DBX94.0	DB390x.DBX4002.0
Heat sink temperature prewarning	DB31, ...DBX94.1	DB390x.DBX4002.1
Run-up completed	DB31, ...DBX94.2	DB390x.DBX4002.2
$\|M_d\| < M_{dx}$	DB31, ...DBX94.3	DB390x.DBX4002.3
$\|n_{act}\| < n_{min}$	DB31, ...DBX94.4	DB390x.DBX4002.4
$\|n_{act}\| < n_x$	DB31, ...DBX94.5	DB390x.DBX4002.5
$n_{act} = n_{set}$	DB31, ...DBX94.6	DB390x.DBX4002.6
Motor/drive data set interface: Formatting	DB31, ...DBX130.0-4	DB390x.DBX4008.0-4
Motor/drive data set interface: Valid	DB31, ... DBX130.7	DB390x.DBX4008.7

A3: Axis Monitoring, Protection Zones 2

2.1 Brief description

2.1.1 Axis monitoring functions

Comprehensive monitoring functions are present in the controller for protection of people and machines:

- Contour monitoring
- Positioning monitoring
- Zero-speed monitoring
- Clamping monitoring
- Speed-setpoint monitoring
- Actual-velocity monitoring
- Measuring system monitoring
- Limit-switch monitoring
- Monitoring of the working area limitation

2.1.2 Protection zones

With the help of protection zones, elements of the machine (e.g. spindle chuck, tool changer, toolholder, tailstock, movable probe, etc.) and the workpiece can be protected against collisions.

During automatic execution of part programs in the AUTOMATIC or MDA mode, the NC checks at the start of every part program block whether a collision between protection zones can occur upon moving along the programmed path.

After manual deactivation of an active protection zone, traversing can be performed in this zone. After leaving the protection zone, the protection zone automatically becomes active again.

The definition, activation and deactivation of protection zones takes place via part program instructions.

2.2 Axis monitoring functions

2.2.1 Contour monitoring

2.2.1.1 Contour error

Contour errors are caused by signal distortions in the position control loop.

Signal distortions can be linear or non-linear.

Linear signal distortions

Linear signal distortions are caused by:

- Speed and position controller not being set optimally
- Different servo gain factors of the feed axes involved in creating the path

  With the same servo gain factor for two linear-interpolated axes, the actual position follows the set position along the same path but with a time delay. With different servo gain factors, a parallel offset arises between the set and actual path.
- Unequal dynamic response of the feed drives

  Unequal drive dynamic responses lead to path deviations especially on contour changes. Circles are distorted into ellipses by unequal dynamic responses of the two feed drives.

Non-linear signal distortions

Non-linear signal distortions are caused by:

- Activation of the current limitation within the machining area
- Activation of the limitation of the speed setpoint
- Backlash within and/or outside the position control loop

  When traversing a circular path, contour errors occur primarily due to the reversal error and friction.

  During motion along straight lines, a contour error arises due to a reversal error outside the position control loop, e.g. due to a tilting milling spindle. This causes a parallel offset between the actual and the set contour. The shallower the gradient of the straight line, the larger the offset.
- Nonlinear friction behavior of slide guides

2.2.1.2 Following-error monitoring

Function

In control engineering terms, traversing along a machine axis always produces a certain following error, i.e. a difference between the set and actual position.

The following error that arises depends on:

- Position control loop gain

  MD32200 $MA_POSCTRL_GAIN (servo gain factor)

- Maximum acceleration

  MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)

- Maximum velocity

  MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

- With activated feedforward control:

  Precision of the path model and the parameters:

  MD32610 $MA_VELO_FFW_WEIGHT (factor for the velocity feedforward control)

  MD32800 $MA_EQUIV_CURRCTRL_TIME (equivalent time constant current control loop for feedforward control)

  MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

In the acceleration phase, the following error initially increases when traversing along a machine axis. After a time depending on the parameterization of the position control loop, the following error then remains constant in the ideal case. Due to external influences, more or less large fluctuations in the following error always arise during a machining process. To prevent these fluctuations in the following error from triggering an alarm, a tolerance range within which the following error may change must be defined for the following-error monitoring:

MD36400 $MA_CONTOUR_TOL (Contour monitoring tolerance range)

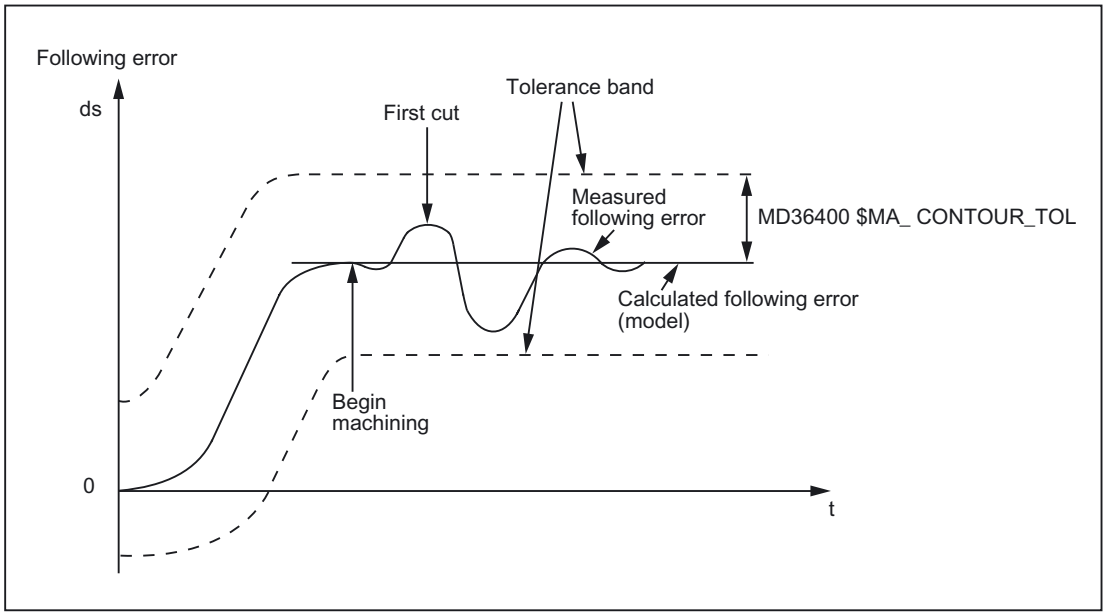


Figure 2-1 Following-error monitoring

Effectiveness

The following-error monitoring only operates with active position control and the following axis types:

- Linear axes with and without feedforward control
- Rotary axes with and without feedforward control
- Position-controlled spindles

Fault

If the configured tolerance limit is exceeded, the following alarm appears:

25050 "Axis <Axis identifier> Contour monitoring"

The affected axis/spindle is stopped via the configured braking ramp in follow-up mode:

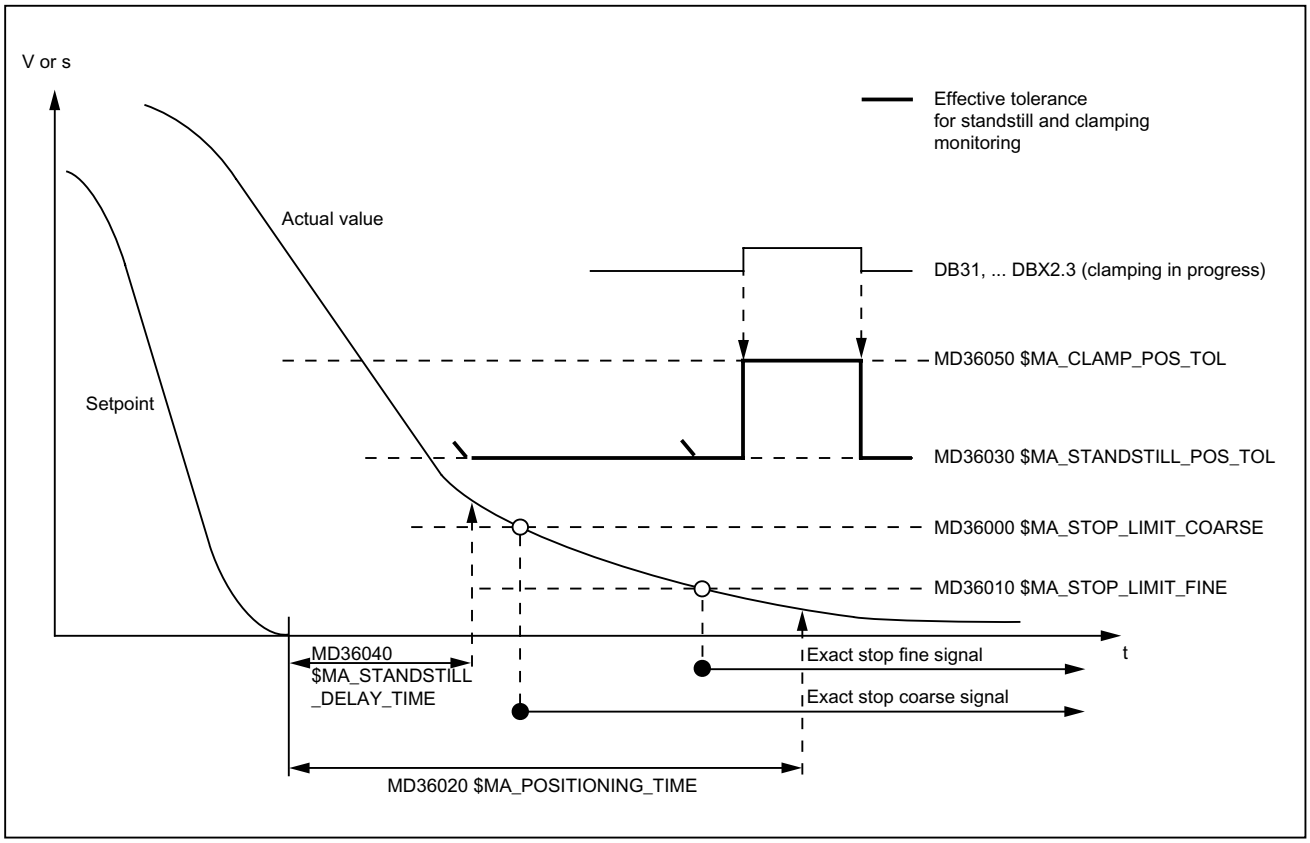
MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

2.2.2 Positioning, zero speed and clamping monitoring

2.2.2.1 Correlation between positioning, zero-speed and clamping monitoring

Overview

The following overview shows the correlation between the positioning, zero speed and clamping monitoring functions:



2.2.2.2 Positioning monitoring

Function

At the end of a positioning operation:

- Set velocity = 0 **AND**
- DB31, ... DBX64.6/7 (motion command minus/plus) = 0

checks the position monitoring to ensure that the following error of every participating machine axis is smaller than the exact-stop fine tolerance during the delay time.

MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)

MD36020 $MA_POSITIONING_TIME (delay time exact stop fine)

After reaching "Exact stop fine", the position monitoring is deactivated.

Note

The smaller the exact stop fine tolerance is, the longer the positioning operation takes and the longer the time until block change.

Rules for MD setting

MD36010 $MA_STOP_LIMIT_FINE	MD36020 $MA_POSITIONING_TIME
Large	Can be selected relatively short
Small	Must be selected relatively long

MD32200 $MA_POSCTRL_GAIN (servo gain factor)	MD36020 $MA_POSITIONING_TIME
Small	Must be selected relatively long
Large	Can be selected relatively short

Effectiveness

The position monitoring only operates with active position control and the following axis types:

- Linear axes
- Rotary axes
- Position-controlled spindles

Fault

If the configured position-monitoring time is exceeded, the following alarm appears:

25080 "Axis <Axis identifier> Position monitoring"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

2.2.2.3 Zero-speed monitoring

Function

At the end of a positioning operation:

- Set velocity = 0 **AND**
- DB31, ... DBX64.6/7 (motion command minus/plus) = 0

checks the zero-speed monitoring to ensure that the following error of every participating machine axis is smaller than the standstill tolerance during the delay time.

MD36040 $MA_STANDSTILL_DELAY_TIME (zero-speed monitoring delay time)

MD36030 $MA_STANDSTILL_POS_TOL (standstill tolerance)

After reaching the required exact-stop state, the positioning operation is completed:

DB31, ... DBX60.6/7 (position reached with exact stop coarse/fine) = 1

The position-monitoring function is deactivated and is replaced by the zero-speed monitoring.

Zero-speed monitoring monitors the adherence to the standstill tolerance. If no new travel request is received, the machine axis must not depart from the standstill tolerance.

Effectiveness

The zero-speed monitoring only operates with active position control and the following axis types:

- Linear axes
- Rotary axes
- Position-controlled spindles

Fault

If the delay time and/or the standstill tolerance is exceeded, the following alarm appears:

25040 "Axis <Axis identifier> Zero-speed monitoring"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

2.2.2.4 Parameter set-dependent exact stop and standstill tolerance

For adaptation to different machining situations and/or axis dynamics, e.g.:

- Operating state A: High precision, long machining time
- Operating state B: Lower precision, shorter machining time
- Changing of the mass relationships after gear change

the positioning tolerances:

- MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)
- MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)
- MD36030 $MA_STANDSTILL_POS_TOL (standstill tolerance)

can be weighted with a common factor depending on the parameter set:

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

Because the factor applies in common for all three position tolerances, the relationship between the values remains constant.

2.2.2.5 Clamping monitoring

Function

For machine axes that are mechanically clamped upon completion of a positioning operation, larger motions can result from the clamping process (> standstill tolerance). As a result, zero-speed monitoring is replaced by clamping monitoring during the clamping process.

Clamping monitoring monitors the adherence to the configured clamping tolerance:

MD36050 $MA_CLAMP_POS_TOL (clamping tolerance)

Activation

The clamping monitoring is activated by the following interface signal:

DB31, ... DBX2.3 (clamping in progress)

Note

The clamping monitoring is not active in "follow-up mode" (DB31, ... DBX1.4 = 1).

Fault

If the clamping tolerance is exceeded, the following alarm appears:

26000 "Clamping monitoring"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

Automatic stop to release the clamping

If a clamped axis must be traversed again in continuous-path mode, the NC stops the path motion for Look Ahead at the start of the motion block of the clamped axis until the clamped axis can once again be traversed. If the clamping is released before stopping, the path motion is not stopped.

Parameterization:

MD36052 $MA_STOP_ON_CLAMPING = 'H01' (special function for clamped axis)

Note

The NC detects whether an axis is clamped based on the "servo enable" state of the axis:

DB31, ... DBX2.2 = 0: No servo enable ⇒ axis is clamped

DB31, ... DBX2.2 = 1: Servo enable ⇒ axis is not clamped

Requirements for the PLC user program

- The axis is always removed from the clamp when a travel command is pending.
- The following is always valid for the axis:

  DB31, ... DBX2.2 (servo enable) = 0: Axis is clamped.

  DB31, ... DBX2.2 (servo enable) = 1: Axis is not clamped.

The following figure shows an example of the interface signals and states upon releasing of the axis clamp:

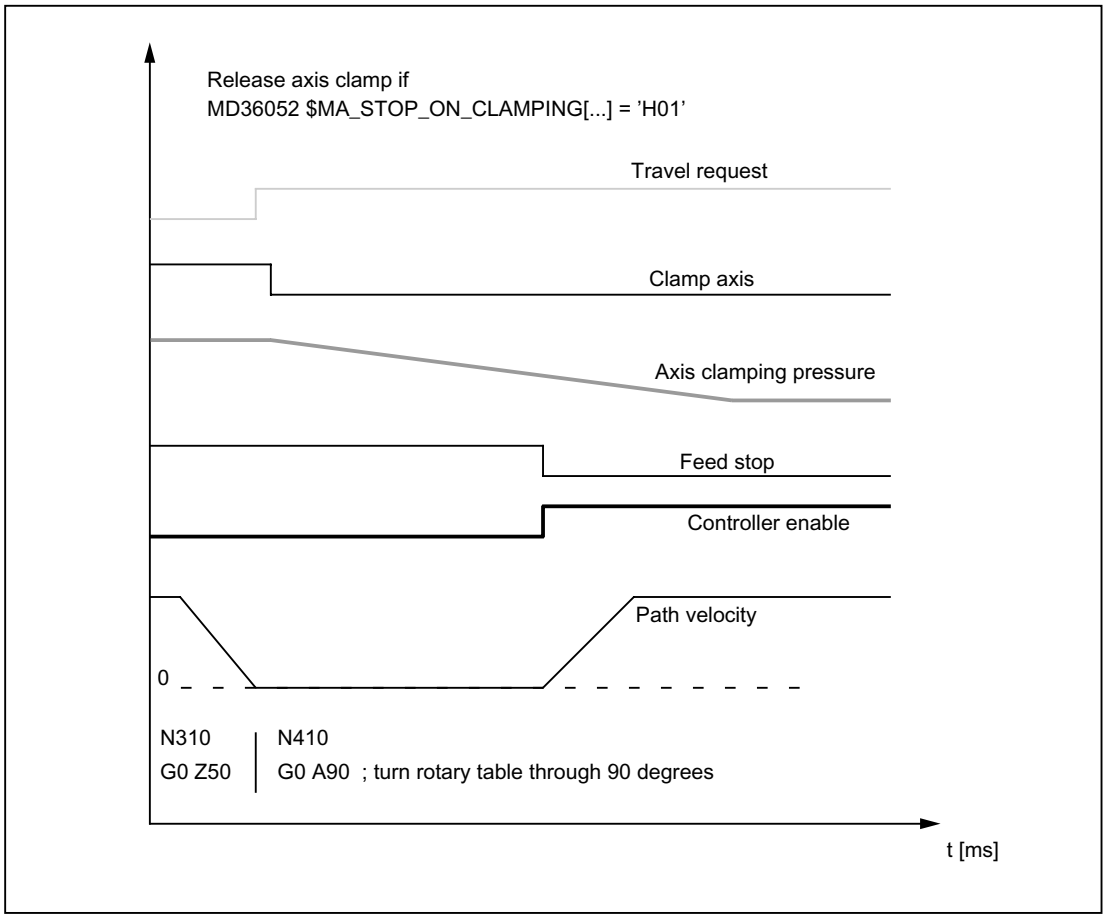


Figure 2-2 Release axis clamp if MD36052 $MA_STOP_ON_CLAMPING = 'H01'

The part program blocks N310 and N410 refer to the following programming example:

```
Program code                                    Comment
N100     G0 X0 Y0 Z0 A0 G90 G54 F500
N101     G641 ADIS=.1 ADISPOS=5
N210     G1 X10                                 ; Machining
N220     G1 X5 Y20
N310     G0 Z50                                 ; Retraction
N410     G0 A90                                 ; Turn rotary table
N510     G0 X100                                ; Approach
N520     G0 Z2
N610     G1 Z-4                                 ; Machining
N620     G1 X0 Y-20
```

Optimized release of the axis clamping via travel command

If a clamped axis is to be traversed in continuous-path mode, a travel command is issued for the clamped axis in the rapid traverse blocks (G0) immediately before the traversing block of the clamped axis. This way, the PLC user program can release the axis clamp again in time.

Note

The travel command is set a maximum of two rapid traverse blocks prior (including intermediate blocks) to retain the reference to the initiating part program block.

Parameterization:

MD36052 $MA_STOP_ON_CLAMPING = 'H03' (special function for clamped axis)

Requirements for the PLC user program

- The axis is removed from the clamp as soon as a travel command is pending.
- The axis may be removed from the clamping even when only positioning is being performed (G0).

The following figure shows an example of the interface signals and states upon releasing of the axis clamp:

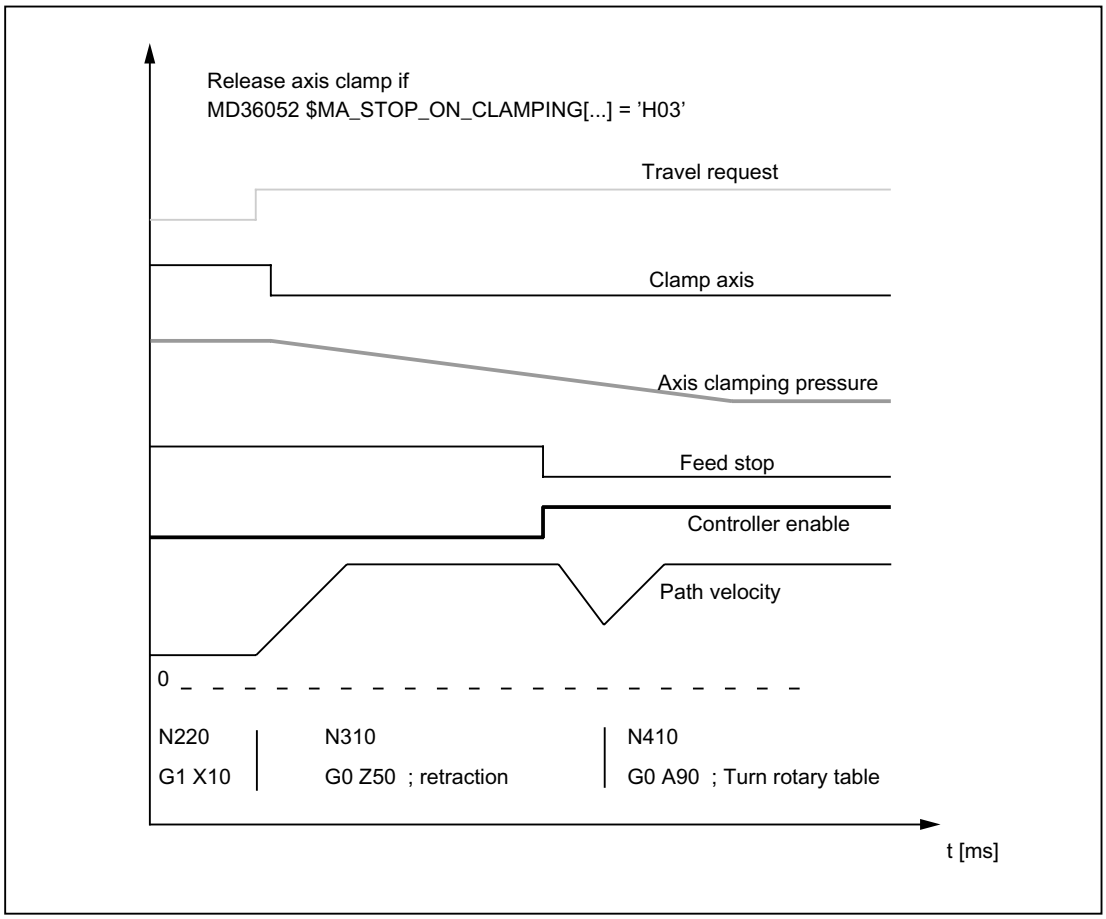


Figure 2-3 Release axis clamp if MD36052 $MA_STOP_ON_CLAMPING = 'H03'

Automatic stop to set the clamping

If an axis is to be clamped in continuous-path mode, the NC stops the path motion before the next "Non-rapid traverse block" if the axis has not been clamped by then, i.e. the PLC has set the feedrate override value to zero.

Parameterization:

MD36052 $MA_STOP_ON_CLAMPING = 'H04' (special function for clamped axis)

Requirements for the PLC user program

- The axis is always clamped when no travel command is pending.
- The axis does not have to be clamped during positioning of the other axes.

  It can be seen whether the axes are being positioned depending on whether rapid traverse (G0) is programmed.

  The stop command is therefore not set immediately at the beginning of the block containing the axis, but at the beginning of the next machining block (traversing block that is not traversed with rapid traverse).
- The axis is clamped if the feedrate override of a machining block is not equal to 0.

  If the axis is clamped before the next machining block, i.e. the feedrate override is other than 0 again, no stop is generated.

The following figure shows an example of the interface signals and states upon setting of the axis clamp. The part program blocks N410, N510, N520 and N610 refer to the schematic example under supplementary conditions.

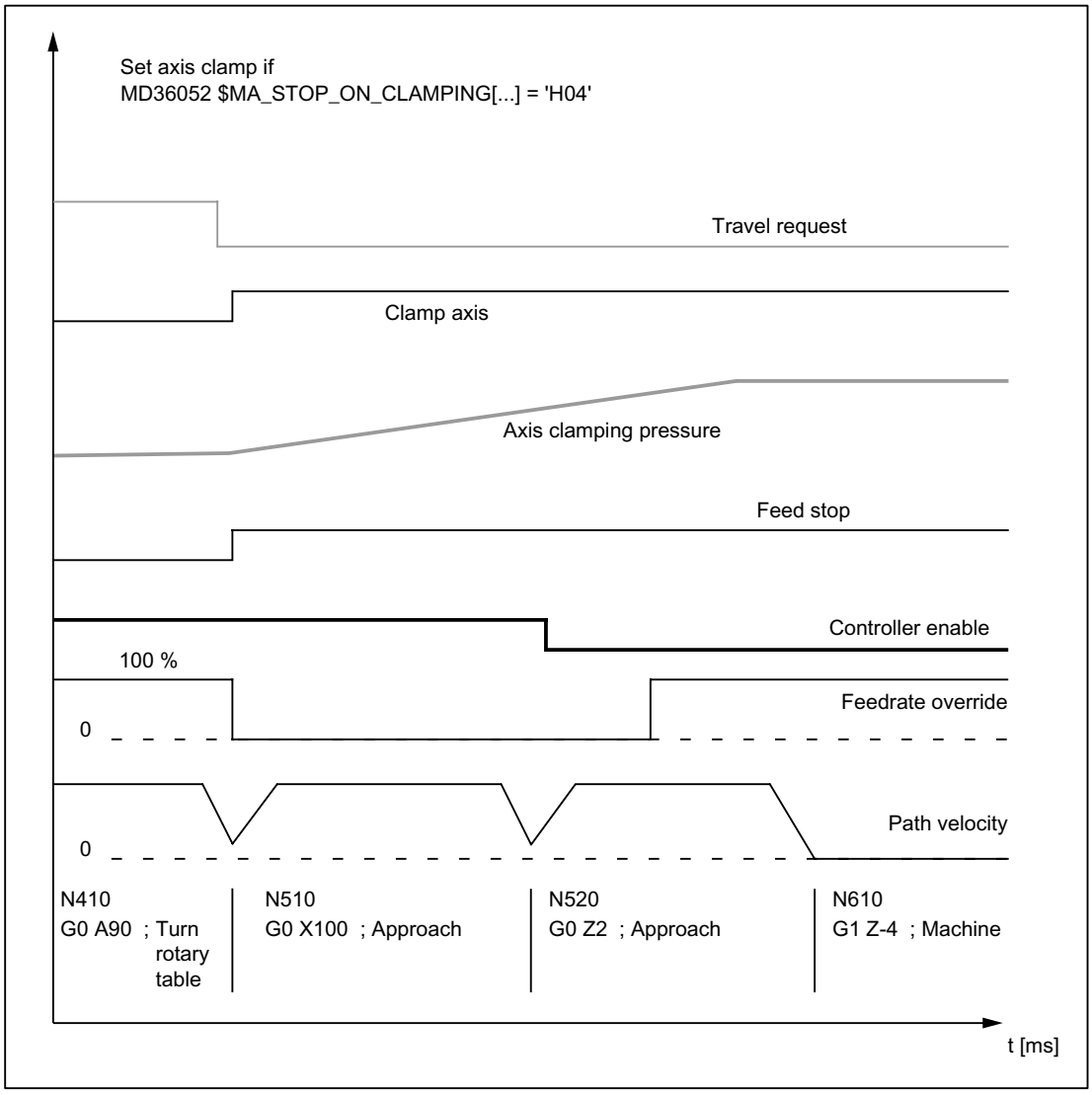


Figure 2-4 Set axis clamp if MD36052 $MA_STOP_ON_CLAMPING = 'H04'

Supplementary conditions

Continuous-path mode

For the above-mentioned functions:

- Automatic stop to release the clamping
- Optimized release of the axis clamping via travel command
- Automatic stop to set the clamping

the "Look Ahead" function must be active.

Part program blocks without path motion (e.g. M82/M83) interrupt continuous-path mode and thus also the "Look Ahead" function.

Example:

The part program blocks N320 and N420 are inserted in the programming example used.

```
Program code                                    Comment
N100     G0 X0 Y0 Z0 A0 G90 G54 F500
N101     G641 ADIS=.1 ADISPOS=5
N210     G1 X10                                 ; Machining
N220     G1 X5 Y20
N310     G0 Z50                                 ; Retraction
N320     M82                                    ; No path motion
N410     G0 A90                                 ; Turn rotary table
N420     M83                                    ; No path motion
N510     G0 X100                                ; Approach
N520     G0 Z2
N610     G1 Z-4                                 ; Machining
N620     G1 X0 Y-20
```

The function behaves as follows:

- MD36052 $MA_STOP_ON_CLAMPING = 'H03'

  No longer has an effect.

  The travel command is set in Look Ahead mode only for blocks with active continuous-path mode. M82 generates a stop and thus interrupts the continuous-path mode. The Look Ahead stopping on N410 would not be necessary because stopping occurs anyway.

- MD36052 $MA_STOP_ON_CLAMPING = 'H04'

  Generates a stop irrespective of M83 which is executed as a function of "feedrate override 0%". The axis is thus stopped before the first machining block.

Note

MD36052 $MA_STOP_ON_CLAMPING = 'H01' or 'H04'

Both functions can be used irrespective of the clamping of axes:

- MD36052 $MA_STOP_ON_CLAMPING = 'H01'

  Generates a Look Ahead stop for the path motion if no servo enable signal is active for the relevant axis.

- MD36052 $MA_STOP_ON_CLAMPING = 'H04'

  Generates a Look Ahead stop for the path motion if the feedrate override = 0% at the transition from the part program blocks with rapid traverse to part program blocks without rapid traverse.

Both functions ensure that the path motion in continuous-path mode is already stopped before the start of the relevant part program block and not just within the part program block.

Block change criterion: Clamping tolerance

After activation of clamp monitoring:(DB31, ... DBX2.3 = 1), the block change criterion for traversing blocks in which the axis stops at the end of the block no longer acts as the corresponding exact-stop condition, but the configured clamping tolerance:

MD36050 $MA_CLAMP_POS_TOL (clamping tolerance with interface signal "Clamping active")

Behavior upon releasing of the clamp

If the axis was moved by the clamping process, it is returned by the NC to the position setpoint after releasing of the clamp and setting of the servo enable state. Repositioning depends on whether "Follow-up mode" was activated for the axis:

- Without follow-up mode: Repositioning by position controller
- With follow-up mode: Repositioning by interpolator

See also interface signal DB31, ... DBX1.4 (follow-up mode).

Note

The following interface signals can be evaluated by the PLC user program as the criterion for activation of the "Follow-up mode":

DB31, ... DBX60.6 / 7 (position reached with exact stop coarse / fine)

2.2.3 Speed-setpoint monitoring

Function

The speed setpoint comprises:

- Speed setpoint of the position controller
- Speed setpoint portion of the feedforward control (with active feedforward control only)
- Dift compensation (only for drives with analog setpoint interface)

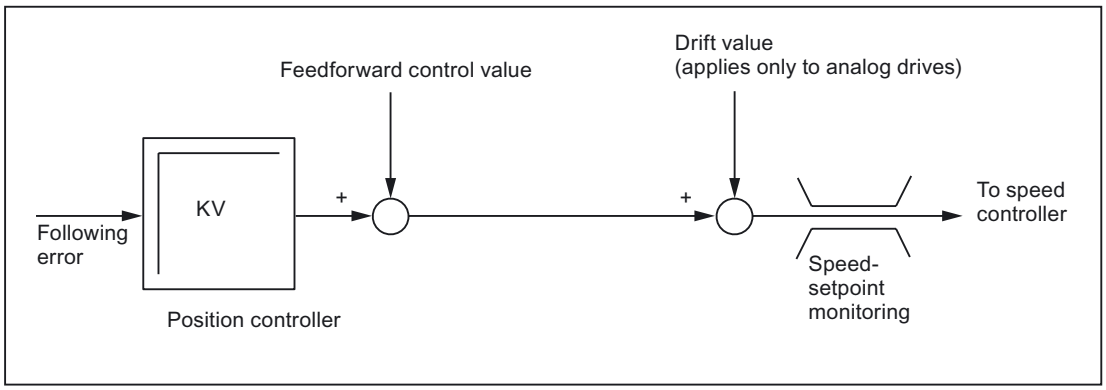


Figure 2-5 Speed setpoint calculation

The speed-setpoint monitoring ensures by limiting the control or output signal (10V for analog setpoint interface or rated speed for digital drives) that the physical limitations of the drives are not exceeded:

MD36210 $MA_CTRLOUT_LIMIT (Maximum speed setpoint)

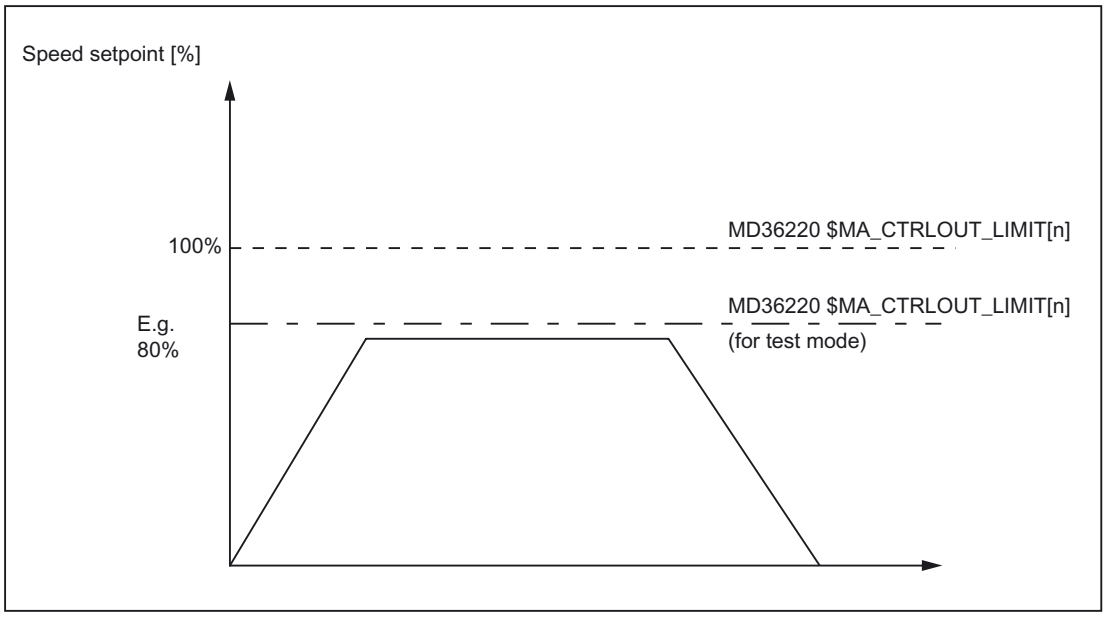


Figure 2-6 Speed setpoint limitation

Speed-setpoint monitoring delay

To prevent an error reaction from occurring in every speed-limitation instance, a delay time can be configured:

MD36220 $MA_CTRLOUT_LIMIT_TIME (Speed-setpoint monitoring delay)

Only if the speed limitation is required for longer than the configured time does the corresponding error reaction occur.

Effectivity

The speed-setpoint monitoring is only active for closed-loop position-controlled axes and cannot be deactivated.

Fault

If the configured delay time is exceeded, the following alarm appears:

25060 "Axis <Axis identifier> Speed-setpoint monitoring"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(Maximum time for braking ramp when an error occurs)

Note

Upon reaching the speed-setpoint monitoring, the position feedback loop of the axis becomes non-linear due to the limitation. Contour errors result if the axis is involved in generating the contour.

2.2.4 Actual-velocity monitoring

Function

The actual-velocity monitoring checks that the current actual velocity of a machine axis/spindle does not exceed the configured threshold:

MD36200 $MA_AX_VELO_LIMIT (velocity-monitoring threshold)

The threshold should be 10-15% above the configured maximum velocity.

- For axes:

  MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

- For spindles:

  MD35110 $MA_GEAR_STEP_MAX_VELO_LIMIT[n] (maximum speed of gear stage)

If you use this setting the speed will not normally exceed the velocity-monitoring threshold (exception: Drive error).

Activation

The actual-velocity monitoring is activated as soon as the active measuring system returns valid actual values (encoder limit frequency not exceeded).

Effectiveness

The actual-velocity monitoring only operates with active position control and the following axis types:

- Linear axes
- Rotary axes
- Open-loop-controlled and position-controlled spindles

Fault

If the threshold is exceeded, the following alarm is displayed:

25030 "Axis <Axis identifier> Actual-velocity alarm limit"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

2.2.5 Measuring system monitoring

The NC has no direct access to the measuring system hardware, therefore measuring system monitoring is mainly performed by the drive software.

Monitoring functions in the drive

- Monitoring of hardware faults (e.g. measuring system failure, wire breakage)
- Zero-mark monitoring

References:
Drive Functions SINAMICS S120

Measuring system monitoring functions carried out in the drive are mapped on the NCK alarms (alarm 25000 and following) or NC reactions (e.g. abort of referencing or on-the-fly measuring). The exact behavior of the NC depends on the setting in the machine data:

MD36310 $MA_ENC_ZERO_MONITORING

Value	Meaning	
= 0	Monitoring of HW faults:	ON If a hardware fault is detected in the active measuring system, POWER ON alarm 25000 is displayed: "Axis <Axis identifier> Hardware fault active encoder" The affected axis is stopped via the configured braking ramp in follow-up mode: MD36610 $MA_AX_EMERGENCY_STOP_TIME (maximum time for braking ramp when a fault occurs) If a hardware fault is detected in the passive measuring system, alarm 25001 is displayed: "Axis <Axis identifier> Hardware fault passive encoder" There is no further alarm response.
	Zero-mark monitoring:	OFF Alarms 25020 and 25021 (see below) are suppressed.
= 100	No zero-mark monitoring as well as hiding of all encoder monitoring functions (i.e. in addition to alarm 25020 (25021)), alarms 25000 (25001) and 25010 (25011) are suppressed.	

Value	Meaning	
> 0 but < 100	Monitoring of HW faults:	ON (see above)
	Zero-mark monitoring:	ON If zero-mark monitoring is tripped in the active measuring system, alarm 25020 is displayed: "Axis <axis identifier> zero-mark monitoring of active encoder" The affected axis is stopped via the configured braking ramp in follow-up mode: MD36610 $MA_AX_EMERGENCY_STOP_TIME (maximum time for braking ramp when a fault occurs) If zero-mark monitoring is tripped in the passive measuring system, alarm 25021 is displayed: "Axis <Axis identifier> Zero-mark monitoring of passive encoder" There is no further alarm response.
> 100	Monitoring of HW faults:	ON with attenuated error message: The POWER ON alarm 25000 is replaced by the reset alarm 25010 and the reset alarm 25001 replaced by the cancel alarm 25011.
	Zero-mark monitoring:	ON (see above)

For details on the alarms, see:

References:
Diagnostics Manual

NOTICE
For hardware faults, the referencing status of the machine axis is reset: DB31, ... DBX60.4/5 (referenced/synchronized 1/2) = 0

Monitoring functions in the NCK

- Encoder-limit-frequency monitoring
- Plausibility check for absolute encoders

2.2.5.1 Encoder-limit-frequency monitoring

Function

The NC encoder-limit-frequency monitoring is based on the configuration and telegram information of the drive. It monitors that the encoder frequency does not exceed the configured encoder limit frequency:

MD36300 $MA_ENC_FREQ_LIMIT (encoder limit frequency)

Encoder-limit-frequency monitoring always refers to the active measuring system selected in the NC/PLC interface:

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

Effectiveness

The encoder limit frequency is operative for:

- Linear axes
- Rotary axes
- Open-loop-controlled and position-controlled spindles

Fault

Upon exceeding of the encoder limit frequency, the following occurs:

- Message to the PLC:

  DB31, ... DBX60.2 or 60.3 = 1 (encoder limit frequency exceeded 1 or 2)

- Spindles

  Spindles are not stopped but continue to turn with speed control.

  If the spindle speed is reduced so much that the encoder frequency passes below the encoder limit frequency, the actual value system of the spindle is automatically resynchronized.

- Axes

  The following alarm is displayed:

  21610 "Channel <Channel number> Axis <Axis identifier> Encoder <Encoder number > Frequency exceeded"

  The affected axis is stopped via the configured braking ramp in follow-up mode:

  MD36610 $MA_AX_EMERGENCY_STOP_TIME
  (maximum time for braking ramp when an error occurs)

Note

If the encoder limit frequency is exceeded, a position-controlled machine axis must be re-referenced (see Section "R1: Referencing (Page 1319)").

2.2.5.2 Plausibility check for absolute encoders

Function

With absolute encoders (MD30240 $MA_ENC_TYPE = 4), absolute values supplied by the measuring system are used to check the plausibility of the actual value.

During the check, the NC compares the cyclic position value held in the position control cycle clock based on the incremental information from the encoder with a new position value generated directly from the absolute and incremental information and checks that the calculated position difference does not exceed the permissible deviation.

MD36310 $MA_ENC_ZERO_MONITORING (permissible deviation in 1/2 coarse increments between the absolute and the incremental encoder track)

Note

The plausibility check of absolute encoders specifically detects all deviations caused by dirt on the absolute track or by faults when transferring the absolute value. However, small errors in the incremental track (burst interference, impulse errors) are not detected. In such instances the plausibility check only responds to deviations in the millimeter range. This form of monitoring should therefore serve as additional monitoring to assist the diagnosis of absolute-position faults.

Note

Rotary absolute encoders

If the plausibility check is to be used for a rotary absolute encoder, the SINAMICS parameter p0979 must be taken into account when setting the modulo range (MD34220 $MA_ENC_ABS_TURNS_MODULO).

NOTICE

Upgrading the NCK software

If the plausibility check is activated in absolute encoders (MD36310 > 0), the existing MD36310 settings must be checked and, if necessary, increased during an upgrade of the NCK software.

Zero mark diagnostics

With absolute encoders, the permissible deviation must be determined for the plausibility check during commissioning. This can be performed via the machine data:

MD36312 $MA_ENC_ABS_ZEROMON_WARNING (zero-mark monitoring warning threshold)

Value	Meaning
0	No zero mark diagnostics
> 0	Permissible deviation in 1/2 coarse increments between the absolute and the incremental encoder track

Procedure when commissioning the system:

1. Deactivate zero-mark monitoring:

   MD36310 $MA_ENC_ZERO_MONITORING = 0

2. Activate zero-mark diagnostics:

   MD36312 $MA_ENC_ABS_ZEROMON_WARNING = 1

3. Move axis and monitor system variable $VA_ENC_ZERO_MON_ERR_CNT (number of detected limit value violations).

4. If $VA_ENC_ZERO_MON_ERR_CNT ≠ 0:

   Increase MD36312 value and repeat step 3.

5. If $VA_ENC_ZERO_MON_ERR_CNT = 0 (over a longer period of time!):

   The correct value for MD36310 is located! Apply the value from MD36312 to MD36310 and then set MD36312 to "0".

Note

Depending on the rigidity of the machine (minimal load masses / moments of inertia are optimum) and the controller settings, the control play "oscillates" with varying degrees of intensity. Account must be taken of this by entering machine-specific limit values in MD36310.

Fault

Alarm 25020

If the plausibility check is tripped in the **active** measuring system, alarm 25020 is displayed:

"Axis <Axis identifier> Zero-mark monitoring of active encoder"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

Alarm 25021

If the plausibility check is tripped in the **passive** measuring system, alarm 25021 is displayed:

"Axis <Axis identifier> Zero-mark monitoring of passive encoder"

There is no further alarm response.

Note

In the event of a fault, the adjustment of the absolute encoder is lost and the axis is no longer referenced. The absolute encoder must be readjusted (see Section "Referencing with absolute encoders (Page 1358)").

NOTICE

Errors in the incremental track that cannot be detected with amplitude monitoring can cause position deviations in the millimeter range. The deviation depends on the lattice pitch/line count and the traversing velocity of the axis when the error occurs.

Complete position monitoring is only possible through redundancy, i.e. through comparison with an independent second measuring system.

2.2.5.3 Customized error reactions

Customized zero-mark monitoring

The default alarm and reaction behavior of the zero-mark monitoring can be adapted in absolute measuring systems (MD30240 $MA_ENC_TYPE = 4) with the aid of system variables. This allows you to perform your own monitoring using a synchronized action or OEM application and to use all of the reaction options available in this application, e.g.:

- Transmit alarm
- Use cycles (e.g. approach tool-change position)
- ...

Example:

Users can adjust the alarm and reaction behavior so that when machining an expensive workpiece, which could be damaged if the axis is stopped as a result of an alarm, machining stops before the machining quality of the workpiece is assessed using appropriate synchronized action commands.

Effectiveness

Customized monitoring can be activated in parallel to or as an alternative to standard zero-mark monitoring, depending on the setting in machine data:

MD36310 $MA_ENC_ZERO_MONITORING

Value	Meaning
0	If only user-specific monitoring is to be implemented, the default zero-mark monitoring must be deactivated: MD36310 = 0 **and** MD36312 = 0
> 0	Customized monitoring and standard zero-mark monitoring operate in parallel.
100	All encoder monitoring functions are deactivated.

If both monitoring functions are active (MD36310 > 0), you can perform **cascaded monitoring**.

Example:

If a value falls below the threshold specified in MD36310, customized monitoring triggers a prewarning; standard zero-marking monitoring will only detect a fault if the threshold is exceeded and will then deactivate automatically.

System variables

You can implement customized error reactions using the following system variables:

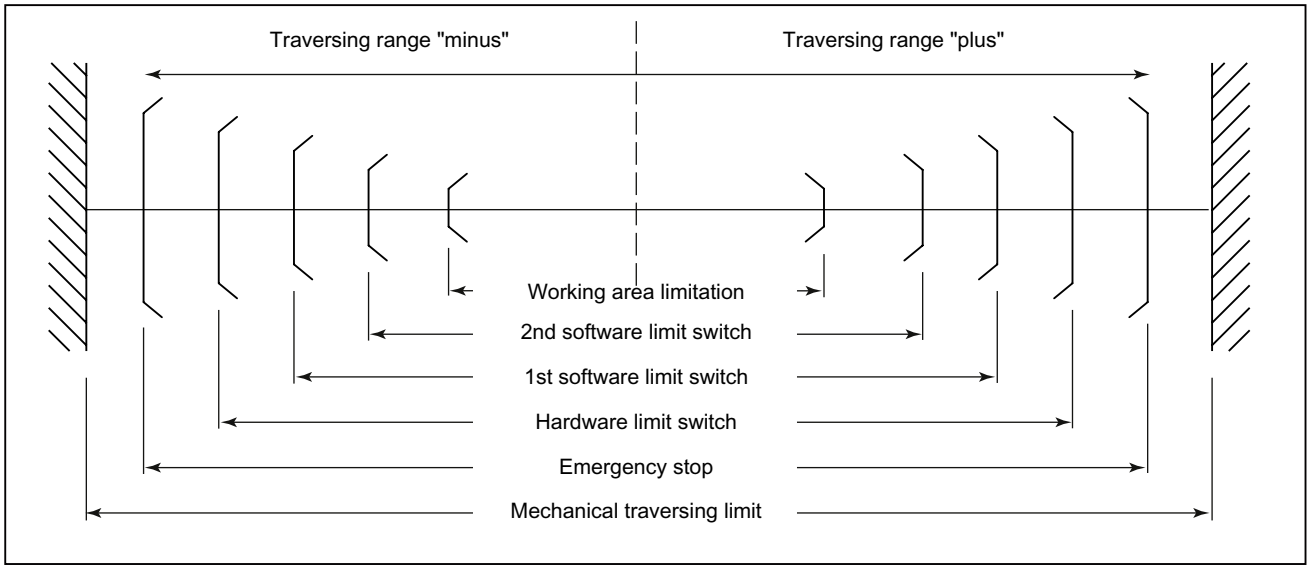
System variable	Meaning
$VA_ENC_ZERO_MON_ERR_CNT[<n>,<axis>]	Number of detected limit value violations. Contains the current number of detected limit value violations when comparing the absolute and the incremental encoder tracks. The value is reset to 0 at: • POWER ON • Selection/deselection of parking Reset does not cause a reset.
$VA_ABSOLUTE_ENC_DELTA_INIT[<n>,<axis>]	Initial difference for absolute encoders. Contains the initial difference between the last buffered absolute position in the static NC memory and the current absolute position. Format of the difference value: Number of internal increments (see MD10200 $MN_INT_INCR_PER_MM or MD10210 $MN_INT_INCR_PER_DEG) The value is updated at: • POWER ON • Warm restart • Deselection of parking • Return below the encoder limit frequency There is no reset at reset.

<n>: Encoder number

<axis>: Axis identifier

2.2.6 Limit-switch monitoring

Overview of the end stops and possible limit-switch monitoring:



2.2.6.1 Hardware limit switch

Function

A hardware limit switch is normally installed at the end of the traversing range of a machine axis. It serves to protect against accidental overtravelling of the maximum traversing range of the machine axis while the machine axis is not yet referenced.

If the hardware limit switch is triggered, the PLC user program created by the machine manufacturer sets the corresponding interface signal:

DB31, ... DBX12.0/1 = 1 (hardware limit switch minus/plus)

Parameterization

The braking behavior of the machine axis upon reaching the hardware limit switch is configurable via the machine data:

MD36600 $MA_BRAKE_MODE_CHOICE (braking behavior on hardware limit switch)

Value	Meaning
0	Braking with the configured axial acceleration
1	Rapid stop (set velocity = 0)

Effectiveness

The hardware limit-switch monitoring is active after the controller has ramped up in all modes.

Effect

Upon reaching the hardware limit switch, the following occurs:

- Alarm 21614 "Channel <Channel number> Axis <Axis identifier> Hardware limit switch <Direction>"
- The machine axis is braked according to the configured braking behavior.
- If the axis/spindle is involved in interpolation with other axes/spindles, these are also braked according to their configured braking behavior.
- The traversing keys of the affected machine axis are blocked based on the direction.

2.2.6.2 Software limit switch

Function

Software limit switches serve to limit the traversing range of a machine axis. Per machine axis and per traversing direction, two (1st and 2nd) software limit switches are available:

MD36100 POS_LIMIT_MINUS (1st software limit switch minus)

MD36110 POS_LIMIT_PLUS (1st software limit switch plus)

MD36120 POS_LIMIT_MINUS2 (2nd software limit switch minus)

MD36130 POS_LIMIT_PLUS2 (2nd software limit switch plus)

By default, the 1st software limit switch is active. The 2nd software limit switch can be activated for a specific direction with the PLC user program:

DB31, ... DBX12.2 / 12.3 (2nd software limit switch minus/plus)

Effectiveness

The software limit switches are active:

- Immediately after the successful referencing of the machine axis.
- In all operating modes.

Supplementary conditions

- The software limit switches refer to the machine coordinate system.
- The software limit switches must be inside the range of the hardware limit switches.
- The machine axis can be moved to the position of the active software limit switch.
- PRESET

  After use of the function PRESET, the software limit-switch monitoring is no longer active. The machine must first be re-referenced.
- Endlessly rotating rotary axes

  No software limit-switch monitoring takes place for endlessly rotating rotary axes:

  MD30310 $MA_ROT_IS_MODULO == 1 (modulo conversion for rotary axis and spindle)

  Exception: Setup-rotary axes

Effects

Automatic operating modes (AUTOMATIC, MDA)

- Without transformation, without overlaid motion, unchanged software limit switch:

  A part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is not started.
- With transformation:

  Different reactions occur depending on the transformation type:

  – Behavior as above.

    or

  – The part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is started. The affected machine axis stops at the active software limit switch. The other machine axes participating in the traversing motion are braked. The programmed contour is left during this process.
- With overlaid motion

  The part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is started. Machine axes that are traveling with overlaid motion or have traveled with overlaid motion stop at the active software limit switch in question. The other machine axes participating in the traversing motion are braked. The programmed contour is left during this process.

Manual operating modes

- JOG without transformation

  The machine axis stops at the software limit switch position.

- JOG with transformation

  The machine axis stops at the software limit switch position. Other machine axes participating in the traversing motion are braked. The preset path is left during this process.

General

- Changing of the software limit switch (1st ↔ 2nd software limit switch)

  If the actual position of the machine axis after changing lies behind the software limit switch, it is stopped with the maximum permissible acceleration.

- Overrunning the software limit switch in JOG mode

  If the position of the software limit switch is reached and renewed pressing of the traversing button should cause further travel in this direction, an alarm is displayed and the axis is not traversed farther:

  Alarm 10621 "Channel <Channel number> Axis <Axis identifier> is at the software limit switch <Direction>"

2.2.7 Monitoring of the working area limitation

2.2.7.1 General

Function

The "working area limitation" function can be used to limit the traversing range of a channel's geometry and special axes to a permissible operating range. The function monitors compliance with working area limits both in AUTOMATIC mode and in JOG mode.

The following versions are available:

- Working area limitation in the Basic Coordinate System (BCS)

  The traversing range limits are specified relative to the Basic Coordinate System.

- Working area limitation in the workpiece coordinate system (WCS) or adjustable zero system (AZS)

  The traversing range limits are specified relative to the workpiece coordinate system or to the adjustable zero system.

The two types of monitoring are independent of each other. If they are both active at the same time, the traversing range limit which most restricts the access will take effect, depending on the direction of travel.

Reference point at the tool

Taking into account the tool data (tool length and tool radius) and therefore the reference point at the tool when monitoring the working area limitation depends on the status of the transformation in the channel:

- **Transformation inactive**

  Without transformations during traversing motion with an active tool the position of the tool tip P is monitored, i.e. during the monitoring the tool length is considered automatically.

  Consideration of the tool radius must be activated separately:

  MD21020 $MC_WORKAREA_WITH_TOOL_RADIUS (Consideration of the tool radius in the working area limitation)

- **Transformation active**

  In the case of certain transformations the monitoring of the working area limitation may differ from the behavior without transformation:

  – The tool length is a component of the transformation ($MC_TRAFO_INCLUDES_TOOL_X = TRUE):

    In this case the tool length is not considered, i.e. the monitoring refers to the tool carrier reference point.

  – Transformation with change in orientation:

    In the case of transformations with changes in orientation, monitoring is always based on the tool center point. MD21020 has no influence.

  Note

  The machine data $MC_TRAFO_INCLUDES_TOOL_... is analyzed only in certain transformations. Condition for a possible evaluation is that the orientation of the tool with respect to the base coordinate system cannot be changed by the transformation. With standard transformations, the condition is only fulfilled for the "inclined axis" type of transformation.

Response

Automatic operating modes

- With / without transformation

  The parts program block with a programmed traversing motion that would lead to overrunning of the working area limits is not executed.

- With superimposed motion

  The axis, which would violate the working area limitation due to a superimposed motion, is braked with maximum acceleration and without jerk limits (BRISK), and will come to a stop in the position of the working area limitation. Other axes involved in the movement are braked according to current acceleration behavior (e.g. SOFT). The path correlation may be lost due to different braking accelerations (contour violation).

Manual operating modes

- JOG with / without transformation

  The axis is positioned at the working area limitation and then stopped.

Powerup response

If an axis moves outside the permissible working area when activating the working area limits, it will be immediately stopped with the maximum permissible acceleration.

Overrunning of the working area limitation in JOG mode

In JOG mode, an axis is moved to no further than its working area limit by the control system. When the traverse button is pressed again, an alarm is displayed and the axis does not traverse any further.

Geo-axis replacement

Through the following machine data it is adjustable, whether during geometry axis change the active working area limitation is retained or deactivated:

MD10604 $MN_WALIM_GEOAX_CHANGE_MODE = <value>

<value>	Meaning
0	The working area limitation is deactivated during the geometry axis change.
1	The working area limitation remains activated during the geometry axis change.

2.2.7.2 Working area limitation in BKS

Application

Using the "working area limitation in BKS", the working area of a machine tool is limited so that the surrounding devices (e.g., tool revolver, measuring stations) are protected against damage.

Working area limits

The lower and upper working area limits of each axes are adjusted through setting data or programmed through part program instructions:

Working area limitation through setting data

The adjustments are done through the immediately effective axis-specific setting data:

SD43420 $SA_WORKAREA_LIMIT_PLUS (Working area limitation plus)

SD43430 $SA_WORKAREA_LIMIT_MINUS (Working area limitation minus)

Programmed working area limitation

The programming is done using the G commands:

`G25 X… Y… Z…`	lower working area limitation
`G26 X… Y… Z…`	upper working area limitation

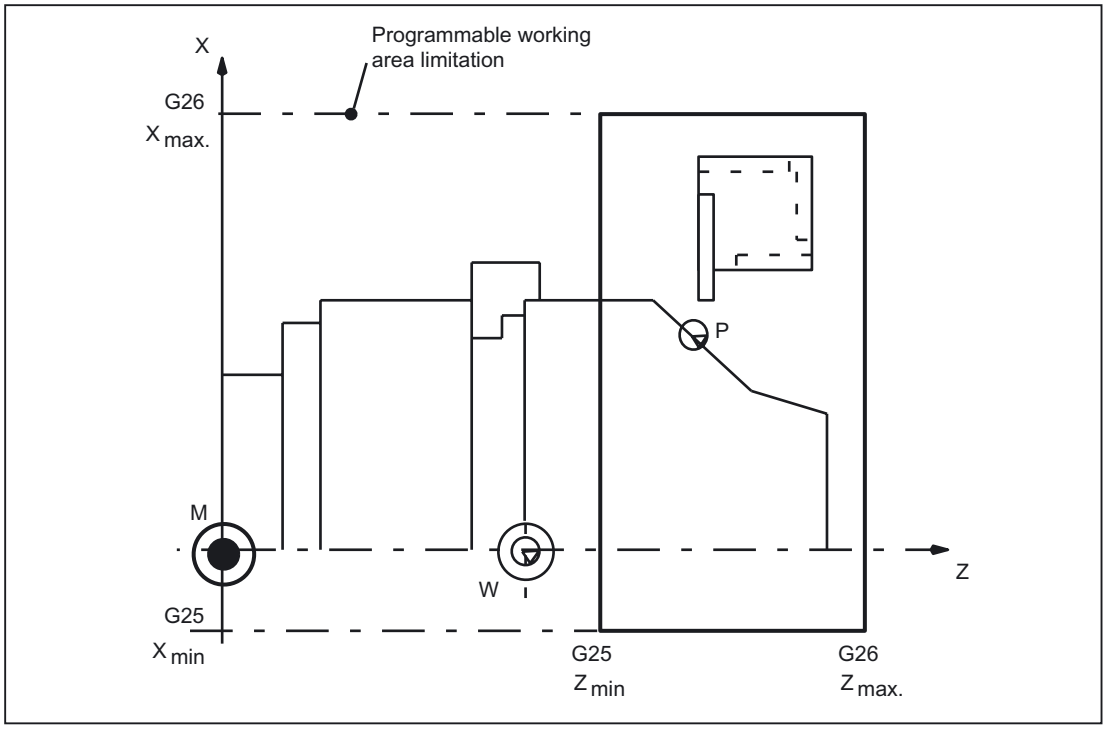


Figure 2-7 Programmed working area limitation

The programmed working area limitation has priority and overwrites the values entered in SD43420 and SD43430.

Activation/Deactivation

Working area limitation through setting data

The activation/deactivation of the working area limitation for each axis takes place in a direction-specific manner via the immediately effective setting data:

SD43400 $SA_WORKAREA_PLUS_ENABLE (Working area limitation active in the positive direction)

SD43410 $SA_WORKAREA_MINUS_ENABLE (Working area limitation active in the negative direction)

Value	Meaning
0	The working area limitation in positive or negative direction is **switched off**.
1	The working area limitation in positive or negative direction is **active**.

Programmed working area limitation

Activation or deactivation of the overall "working area limitation in the BCS" is arranged via part program instructions:

WALIMON	Working area limitation ON
or	
WALIMOF	Working area limitation OFF

Changing the working area limitation

Working area limitation through setting data

HMI user interface: Operating area "Parameter"

- Automatic modes:
  - Changes: possible only in the RESET state
  - Effective: immediately
- Manual operating modes:
  - Changes: always possible
  - Effective: at the start of the next traversing motion

Programmed working area limitation

The working area limitation can be changed in the part program via G25or G26 <Axis identifier> <value>. The change takes effect immediately.

The new working area limitation value is retained after and NC RESET and POWER ON if the back-up process has been activated in the NCK's retentive data storage for SD43420 and SD43430:

MD10710 $MN_PROG_SD_RESET_SAVE_TAB[0] = 43420

MD10710 $MN_PROG_SD_RESET_SAVE_TAB[1] = 43430

Reset position

The reset position for the working area limitation (WALIMON or WALIMOF) is configurable via:

MD20150 $MC_GCODE_RESET_VALUES (RESET position of G groups)

2.2.7.3 Working area limitation in WCS/SZS

Application

The working area limitation enables a channel-specific, flexible workpiece-specific limitation of the traversing range of the channel axes in the workpiece coordinate system (WCS) or settable zero system (SZS). It is intended mainly for use in conventional lathes.

Working area limitation group

In order that the axis-specific working area limits do not have to be rewritten for all channel axes when switching axis assignments, e.g. when switching transformations or the active frame on/off, working area limitation groups are available.

A working area limitation group comprises the following data:

- Working area limits for all channel axes
- Reference system of the working area limitation

The number of the working area limitation groups is set in the machine data:

MD28600 $MC_MM_NUM_WORKAREA_CS_GROUPS

Set working area limits

The working area limits are set channel-specifically for each channel axis via the following system variables:

- $P_WORKAREA_CS_LIMIT_PLUS[<group>, <channel axis name>]
- $P_WORKAREA_CS_LIMIT_MINUS[<group>, <channel axis name>]

With <group> = number of the working area limitation group

Enable working area limits

The working area limits are enabled channel-specifically for each channel axis via the following system variables:

- $P_WORKAREA_CS_PLUS_ENABLE[<group>, <channel axis name>]
- $P_WORKAREA_CS_MINUS_ENABLE[<group>, <channel axis name>]

With <group> = number of the working area limitation group

There is no activation through the enable.

Select reference system

The reference system for a working area limitation group is set via the following system variable for each specific channel:

$P_WORKAREA_CS_COORD_SYSTEM[<group>] = <value>

<value>	Meaning
1	Working area limitation based on the workpiece coordinate system
3	Working area limitation based on the AZS

With <group> = number of the working area limitation group

Activation of working area limits

The working area limits of a working area limitation group are activated in the part program or synchronized action with the command:

`WALCSn (`activation of the working area limits of group n, where n = 1, 2, ...)

Deactivation of working area limits

The working area limits of a working area limitation group are deactivated in the part program or synchronized action with the command:

`WALCS0 (`deactivation of the working area limits active in the channel)

Change working area limits

The working area limits can be changed at any time via the system variables mentioned above. Changes take effect with the next activation of the working area limitation group (`WALCSn`).

Data storage

The system variables of the working area limits are stored retentively in the static memory of the NC.

Note

For the storage of the limiting values for the linear axes, the default setting is considered for the system of units (MD10240 $MN_SCALING_SYSTEM_IS_METRIC).

Data backup

The system variables of the working area limits can be backed up in separate files:

Backup file	For the backup of:
_N_CHx_WAL	Values of the system variables for the channel x.
_N_COMPLETE_WAL	Values of the system variables for all channels.

Note

The system variables of the working area limits are part of the "_N_INITIAL_INI" file.

Behavior in JOG mode

Initial situation:

- In the JOG mode, **several** geometry axes traverse simultaneously (e.g. using several handwheels)
- A **rotating** frame is active between the basic coordinate system (BCS) and the reference coordinate system of the working area limitation (WCS or SZS)

Behavior when a working area limitation responds:

- The traversing motions of the geometry axes that are not affected are continued
- The affected geometry axis is stopped at the working area limit

Set initial setting

The specification of the working area limitation group that is to take effect at power up, reset or end of the part program and start of the part program is performed channel-specifically via the machine data:

MD20150 $MC_GCODE_RESET_VALUE[**59**] = <group>

The working area limitation group that is to take effect is still dependent on the setting in the machine data:

MD20152 $MC_GCODE_RESET_MODE[**59**] = <mode>

<Mode>	Meaning
0	The working area group takes effect in accordance with MD20150
1	The last active working area group remains active

2.2.8 Deactivating all monitoring functions: "Parking"

If a machine axis is brought into the"Parking" state, then for this particular axis, no encoder actual values are acquired, and all of the monitoring functions described in the preceding sections (measuring system, standstill, clamping monitoring, etc.) are deactivated.

Machine axis with measuring system

For a machine axis with measuring system, "parking" is activated by deselecting all measuring systems:

- DB31, ... DBX1.5 = 0 (position measuring system 1)
- DB31, ... DBX1.6 = 0 (position measuring system 2)

When the measuring systems are deactivated, the axis is no longer designated as being referenced:

- DB31, ... DBX60.4 = 0 (referenced / synchronized 1)
- DB31, ... DBX60.5 = 0 (referenced/synchronized 2)

Note

The axis must be re-referenced after the "Park" state has been canceled.

Machine axis without measuring system

For a machine axis without a measuring system (speed-controlled spindle), then a state corresponding to "parking" is activated by withdrawing the controller enable:

- DB31, ... DBX2.1 = 0 (controller enable)

2.3 Protection zones

2.3.1 General

Function

Protection zones are static or moveable in 2- or 3-dimensional ranges within a machine to protect machine elements against collisions.

The following elements can be protected:

- Permanent parts of the machine and attachments (e.g. toolholding magazine, swiveling probe). Only the elements that can be reached by possible axis constellations are relevant.
- Moving parts belonging to the tool (e.g. tool, toolholder)
- Moving parts belonging to the workpiece (e.g. parts of the workpiece, clamping table, clamping shoe, spindle chuck, tailstock).

Protection zones are defined via part program instructions or system variables so that they completely surround the element to be protected. The activation and deactivation of protection zones also takes place via part program instructions.

Protection-zone monitoring by the NC is channelspecific, i.e. all the active protection zones of a channel monitor one another for collisions.

Definition of a protection zone

It is possible to define 2dimensional or 3dimensional protection zones as polygons with a maximum of ten corner points. The protection zones can also contain arc contour elements.

Polygons are defined in a previously defined plane.

Expansion in the 3rd dimension can be limited between - ∞ to + ∞.

The following four cases are possible:

- Dimension of the protection zone from -∞ to +∞
- Dimension of the protection zone from -∞ to the upper limit
- Dimension of the protection zone from the lower limit to +∞
- Dimension of protection zone from lower limit to upper limit.

Coordinate system

The definition of a protection zone takes place with reference to the geometric axis of a channel in the basic coordinate system.

Reference

- Tool-related protection zones

  Coordinates for toolrelated protection zones must be given as absolute values referred to the tool carrier reference point F.

- Workpiece-related protection zones

  Coordinates for workpiecerelated protection zones must be given as absolute values referred to the zero point of the basic coordinate system.

Note

If no toolrelated protection zone is active, the tool path is checked against the workpiece-related protection zones.

If no workpiece-oriented protection zone is active, protection-zone monitoring does not take place.

Orientation

The orientation of the protection zones is determined by the plane definition (abscissa/ordinate), in which the contour is described, and the axis perpendicular to the contour (vertical axis).

The orientation of the protection zones must be the same for the tool and workpiecerelated protection zones.

2.3.2 Types of protection zone

Machine-defined and channel-defined protection zones

- Machine-defined protection zone

  Data for machine-related protection zones are defined once in the control. These protection zones can be activated by all channels.

- Channel-defined protection zones

  Data for channel-related protection zones are defined in a channel. These protection zones can be activated only by this channel.

Example: Doubleslide turning machine

- The toolrelated protection zones are assigned to channel 1 or 2.
- The workpiecerelated protection zones are assigned to the machine.
- The coordinate system must be identical for both channels.

Maximum number of protection areas

The maximum definable number of machine- and channel-related protection zones is set via:

MD18190 $MN_MM_NUM_PROTECT_AREA_NCK (Number of files for machine-related protection zones)

MD28200 $MC_MM_NUM_PROTECT_AREA_CHAN (Number of files for channel-specific protection zones)

Coordinates

The coordinates of a protection zone must always be programmed as absolute values with respect to the reference point of the protection zone. When the protection zone is activated via the part program it is possible to apply a relative offset to the reference point of the protection zone.

Examples

In the following figures some examples for protection zones have been presented:

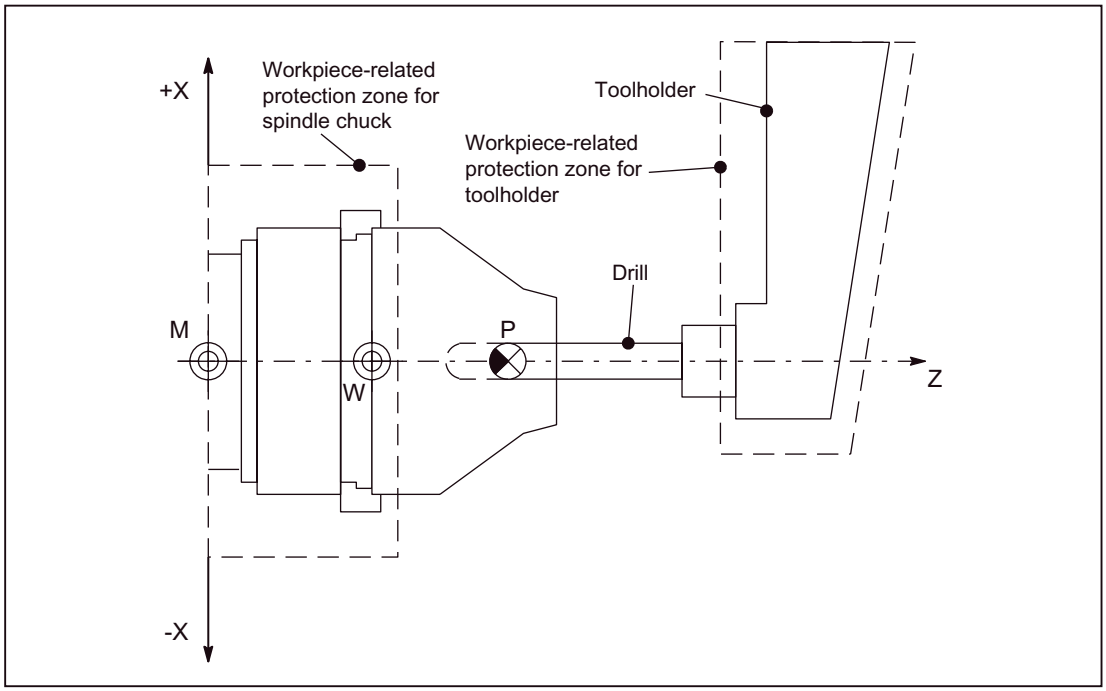


Figure 2-8 Example of application on turning machine

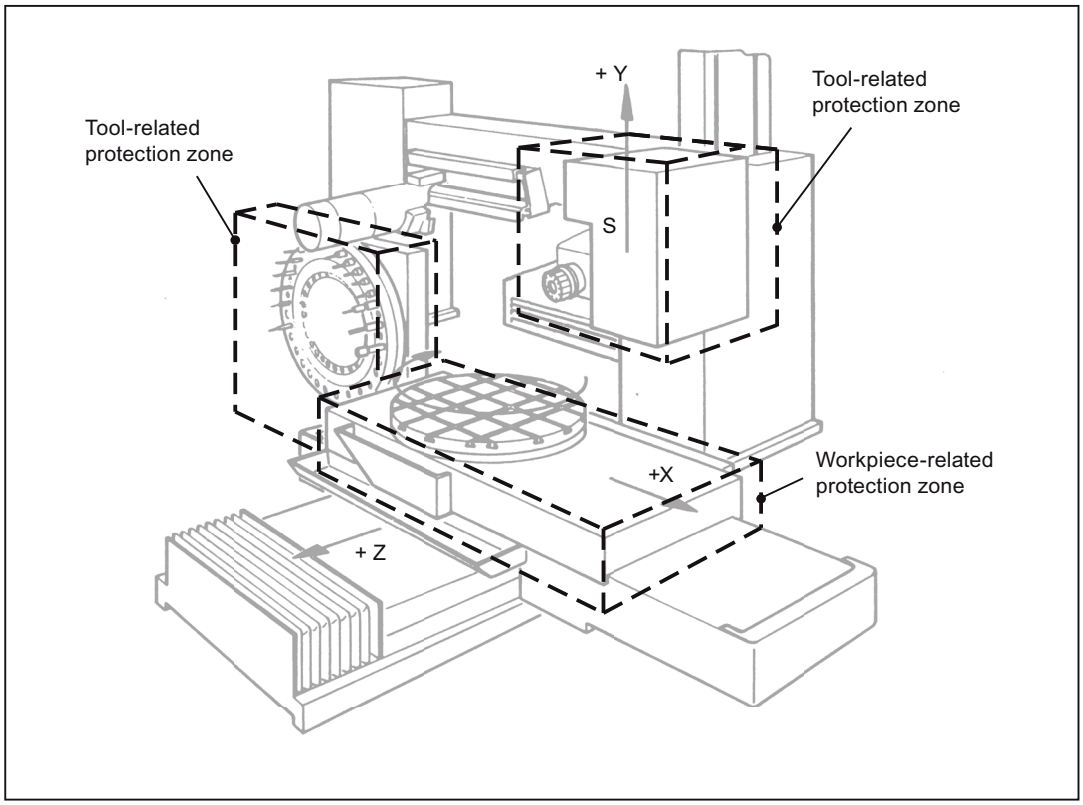


Figure 2-9 Example of a milling machine

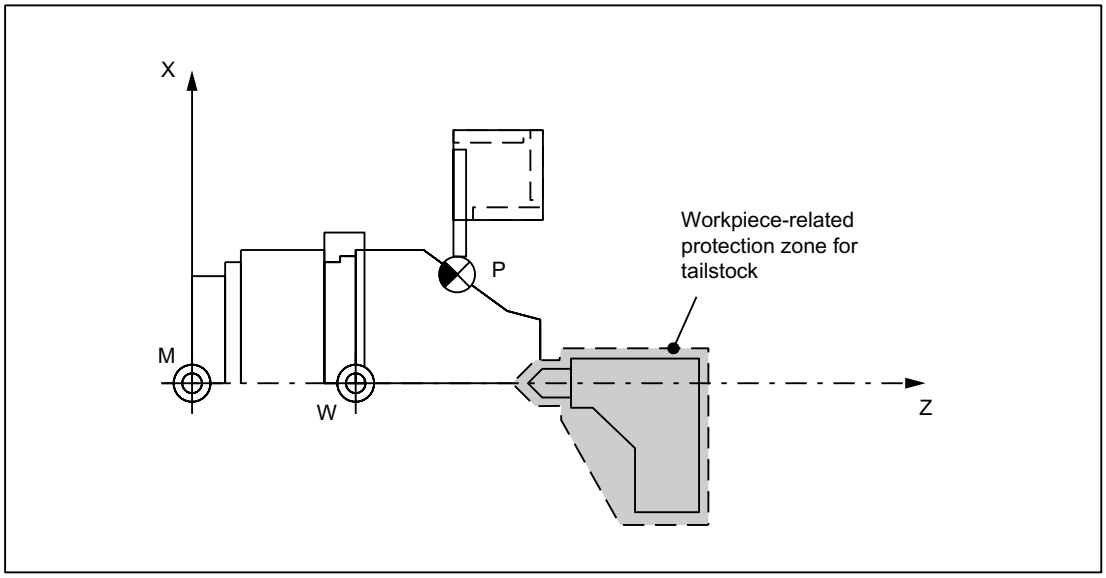


Figure 2-10 Example of a turning machine with relative protection zone for tailstock

2.3.3 Definition via part program instruction

General

A protection-zone definition must contain the following information:

- Protection zone type (workpiece- or tool-related)
- Orientation of the protection zone
- Type of limitation in the third dimension
- Upper and lower limits of the protection zone in the third dimension
- Activation type ("Protection zone immediately active": only possible via system variable)
- Contour elements

Definition of protection zones

The following systematics must be maintained in the definition of protection zones:

- Definition of the working plane: `G17`, `G18` or `G19`
- Definition beginning
  - Channel-specific protection zones: `CPROTDEF(...)`
  - Machine or NC-specific protection zone: `NPROTDEF(...)`
- Contour description for protection zone
- End of definition: `EXECUTE(...)`

Definition of the working plane

The desired working plane to which the contour description of the protection zone refers must be selected with `G17`, `G18`, `G19` before start of the definition. It may not be changed before the end of the definition. Programming of the applicate is not permitted between start and end of the definition.

Definition beginning

The definition start is defined by the corresponding subroutine:

- `CPROTDEF(n, t, applim, appplus, appminus)`
- `NPROTDEF(n, t, applim, appplus, appminus)`

Parameters	Type	Description	
`n`	INT	Number of defined protection zone	
`t`	BOOL	Protection zone type	
		TRUE	Tool-oriented protection zone
		FALSE	Workpiece-related protection zone
`applim`	INT	Type of limitation in the third dimension	
		0	No limitation
		1	Limit in plus direction
		2	Limit in minus direction
		3	Limit in positive and negative direction
`appminus`	REAL	Value of the limit in the negative direction in the 3rd dimension [1]	
`appplus`	REAL	Value of the limit in the positive direction in the 3rd dimension [1]	
[1] The following must be true: `appplus` > `appminus`			

Contour description for protection zone

The contour of a protection zone is described with traversing motions. These are not executed and have no connection to previous or subsequent geometry descriptions. They only define the protection zone.

The contour of a protection zones is specified with up to eleven traversing movements in the selected working plane. The first traversing movement is the movement to the contour. The last point in the contour description must always coincide with the first point of the contour description. In the case of rotationsymmetrical contours (e.g. spindle chuck), the whole contour must be described (not merely the contour to the turning center).

The valid protection zone is the zone left of the contour:

- Internal protection zone

  The contour of an internal protection zone must described in the counterclockwise direction.

- External protection zones (permitted only for workpiece-related protection zones)

  The contour of an external protection zone must be described in the clockwise direction.

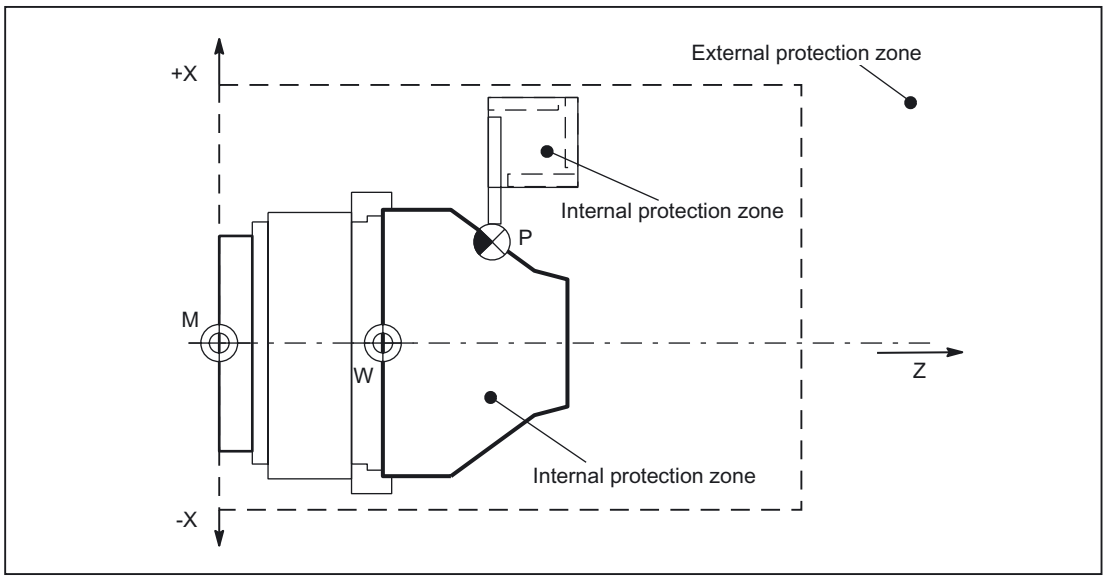


Figure 2-11 Examples: External and internal protection zone

Toolrelated protection zones must be convex. If a concave protection zone is required, the protection zone must be divided up into several convex protection zones.

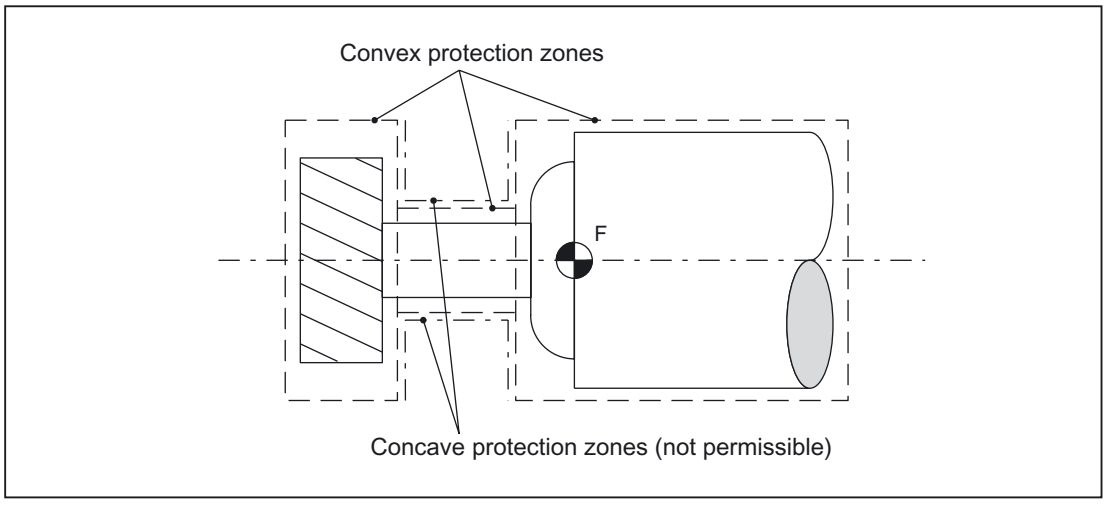


Figure 2-12 Examples: convex and concave tool-related protection zones

Contour elements

The following contour elements are permissible:

- G0, G1 for straight contour elements
- G2 for circle segments in the clockwise direction

  Permissible only for workpiece-related protection zones.

  Not permissible for tool-related protection zones because they must be convex.
- G3 for circular segments in the counterclockwise direction

A protection zone cannot be described by a complete circle. A complete circle must be divided into two half circles.

The sequence G2, G3 or G3, G2 is not permitted. A short G1 block must be inserted between the two circular blocks.

Constraints

During the definition of a protection zone, the following functions must not be active or used:

- Tool radius compensation (cutter radius compensation, tool nose radius compensation)
- Transformation
- Reference point approach (G74)
- Fixed point approach (G75)
- Dwell time (G4)
- Block search stop (STOPRE)
- End of program (M17, M30)
- M functions: M0, M1, M2

Programmable frames (TRANS, ROT, SCALE, MIRROR) and configurable frames (G54 to G57) are ineffective.

Inch/metric switchovers with G70/G71 or G700/G710 are effective.

End of definition

The end of definition is defined by the following subroutine:

```
EXECUTE(NOT_USED)
```

Parameters	Type	Description
NOT_USED	INT	Error variable has no effect in protection zones with EXECUTE.

The definition of a machine-specific or channel-specific protection zone is completed with the subroutine EXECUTE(n).

2.3.4 Definition as per system variable

General

When the protection zones are defined via part program instructions (see Section "Definition via part program instruction (Page 128)"), the protection zone data is stored in system variables. The system variables can also be written directly so that the definition of protection areas can also be performed directly in the system variables.

The same supplementary conditions apply for the definition of the contour of a protection zone as for a protection zone definition via part program instructions.

System variables

The protection zone definitions cover following system variables:

System variable	Type	Meaning	
$SN_PA_ACTIV_IMMED[n] $SC_PA_ACTIV_IMMED[n]	BOOL	Activation type The protection zone is active / not active immediately after the power up of the controller and the referencing of the axes.	
		FALSE	Not immediately active
		TRUE	Immediately active
$SN_PA_T_W[n] $SC_PA_T_W[n]	INT	Protection zone type	
		0	Workpiece-related protection zone
		1	Reserved
		2	Reserved
		3	Tool-related protection zone
$SN_PA_ORI[n] $SC_PA_ORI[n]	INT	Orientation of the protection zone, i.e. polygon definition in the plane of:	
		0	1st and 2nd geometry axis
		1	3rd and 1st geometry axis
		2	2nd and 3rd geometry axis
$SN_PA_LIM_3DIM[n] $SC_PA_LIM_3DIM[n]	INT	Type of limitation in the third dimension	
		0	No limitation
		1	Limit in plus direction
		2	Limit in minus direction
		3	Limit in positive and negative direction
$SN_PA_PLUS_LIM[n] $SC_PA_PLUS_LIM[n]	REAL	Value of the limit in the positive direction in the 3rd dimension	
$SN_PA_MINUS_LIM[n] $SC_PA_MINUS_LIM[n]	REAL	Value of the limit in the negative direction in the 3rd dimension	
$SN_PA_CONT_NUM[n] $SC_PA_CONT_NUM[n]	INT	Number of valid contour elements	
$SN_PA_CONT_TYP[n, i] $SC_PA_CONT_TYP[n, i]	INT	Contour type[i], contour type (`G1`, `G2`, `G3`) of the nth contour element	

System variable	Type	Meaning
$SN_PA_CONT_ABS[n, i] $SC_PA_CONT_ABS[n, i]	REAL	End point of the contour[i], abscissa value
$SN_PA_CONT_ORD[n, i] $SC_PA_CONT_ORD[n, i]	REAL	End point of the contour[i], ordinate value
$SN_PA_CENT_ABS[n, i] $SC_PA_CENT_ABS[n, i]	REAL	Center point of the circular contour[i], absolute abscissa value
$SN_PA_CENT_ORD[n, i] $SC_PA_CENT_ORD[n, i]	REAL	Center point of the circular contour[i], absolute ordinate value
$SN_... are system variables for NC and machine-specific protection zones. $SC_... are system variables for channel-specific protection zones. The index "n" corresponds to the number of the protection zone: 0 = 1. Protection zone The index "i" corresponds to the number of the contour element: 0 = 1. Contour element The contour elements must be defined in ascending order.		

Note

The system variables of the protection zone definitions are not restored with `REORG`.

Data of the protection zone definitions

Data storage

The protection zone definitions are stored in the following files:

File	Blocks
_N_NCK_PRO	Data block for NC-specific protection zones
_N_CHAN1_PRO	Data block for channel-specific protection zones in channel 1
_N_CHAN2_PRO	Data block for channel-specific protection zones in channel 2

Data backup

The protection zone definitions are saved in the following files:

File	Blocks
_N_INITIAL_INI	All data blocks of the protection zones
_N_COMPLETE_PRO	All data blocks of the protection zones
_N_CHAN_PRO	All data blocks of the channel-specific protection zones

2.3.5 Activating and deactivating protection zones

The activation state of a protection zone can have the following values:

- Activated
- Preactivated
- Preactivated with conditional stop
- Deactivated

Activation, preactivation and deactivation via part program

The activation status of a protection zone can be changed in the part program at any time via the following functions:

- Channel-specific protection zone:

```
CPROT(<n>, <state>[, <xMov>, <yMov>, <zMov>])
```

- Machine-specific protection zone:

```
NPROT(<n>, <state>[, <xMov>, <yMov>, <zMov>])
```

Parameters	Type	Description	
<n>:	INT	Protection area number	
<state>:	INT	Activation status	
		0	Deactivated
		1	Preactivated
		2	Activated
		3	Preactivated with conditional stop
<xMov>, <yMov>, <zMov>:	REAL	Additive offset values depending on the reference system: • Workpiece-related protection zone: Machine zero • Tool-related protection zone: Toolholder reference point	

NOTICE
A protection zone is only taken into account after the referencing of all geometry axes of the channel in which it has been activated.

Activation via NC/PLC interface signals

Only protection zones that have been **preactivated** via the part program (see paragraph below "Preactivation via part program") can be activated in the PLC user program via the NC/PLC interface signals:

- DB21, ... DBX8.0 - 9.1 = 1 (activate machine-related protection zone 1 - 10)
- DB21, ... DBX10.0 - 11.1 = 1 (activate channel-specific protection zone 1 - 10)

For preactivation, see paragraph below "Preactivation via part program"

The activation of preactivated protection zones must be performed prior to the traversing motion of the geometry axes. If the activation is performed during the traversing motion, these protection zones are not taken into account in the current traversing motion. Reaction:

- Alarm "10704 Protection zone monitoring is not guaranteed"
- DB21, ... DBX39.0 = 1 (protection zone monitoring not guaranteed)

NOTICE
The activation of preactivated protection zones must be performed prior to the traversing motion of the geometry axes.

Automatic activation after the NC powers up

Protection zones that are to take effect immediately after the NC powers up can be specified via the following system variables:

- Channel-specific protection zones:

  $SC_PA_ACTIV_IMMED[<protection zone number>]

- Machine-specific protection zones:

  $SN_PA_ACTIV_IMMED[<protection zone number>]

Note

With automatic activation, no relative offset of a protection zone is possible.

NOTICE
A protection zone is only taken into account after the referencing of all geometry axes of the channel in which it has been activated.

Preactivation via part program

Protection zones that are to be activated at a later time from the PLC user program must be preactivated in the part program:

```
CPROT or NPROT (<protection zone number>, 1)
```

Preactivated protection zones are displayed via the following NC/PLC interface signals:

- DB21, ... DBX272.0 - 273.1 == 1 (machine-related protection zone 1 - 10 preactivated)
- DB21, ... DBX274.0 - 275.1 == 1 (channel-specific protection zone 1 - 10 preactivated)

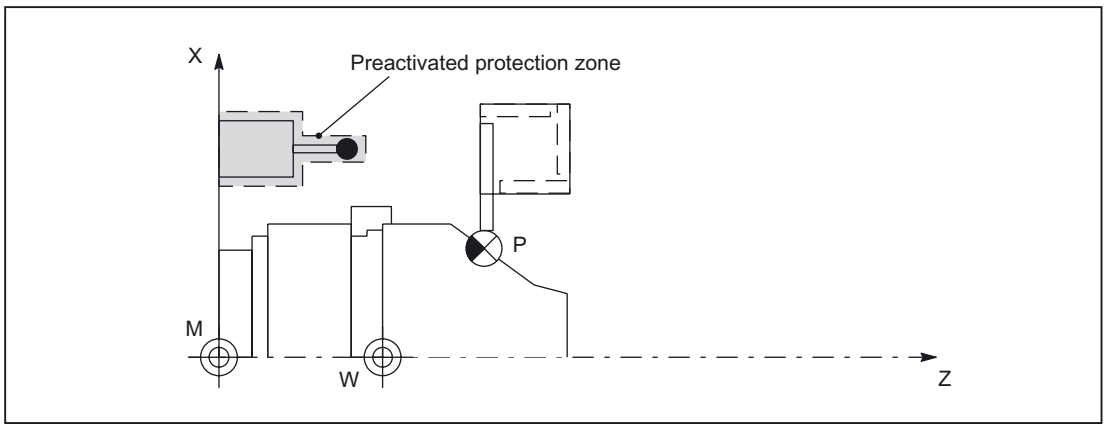


Figure 2-13 Example: Turning machine with preactivated protection zone for a sensor.

Preactivation with conditional stop

The preactivation of protection zones with conditional stop is performed in the part program via:

```
CPROT or NPROT (<protection zone number>, 3)
```

In the case of a preactivated protection zone with conditional stop, a traversing motion is **not** stopped before this if the traversing motion goes into the protection zone. A stop is only performed when the protection zone has been activated. This behavior is to enable uninterrupted machining controlled by the user when the protection zone is only required temporarily.

Note

Possible protection zone violation

If a preactivated protection zone with conditional stop is not activated in good time, the NC may no longer be able to stop before the protection zone in good time because the braking distance after the activation point has not been taken into account.

Deactivation via part program

A protection zone that has been activated via the part program or an NC/PLC interface signal can be deactivated again at any time from the part program with status = 0:

- Channel-specific protection zone:

```
CPROT(n, 0)
```

- Machine-specific protection zone:

```
NPROT(n, 0)
```

Deactivation via NC/PLC interface signal

Only protection zones that have been **preactivated** via a part program and activated via the NC/PLC interface signals, can be deactivated again via the NC/PLC interface signals:

- DB21, ... DBX8.0 to DBX9.1 = 0 (activate machine-related protection zone 1 - 10)
- DB21, ... DBX10.0 to DBX11.1 = 0 (activate channel-specific protection zone 1 - 10)

Protection zones that have been **activated** directly via a part program **cannot** be deactivated from the PLC user program.

Note

It is recommended that protection zones which are to be activated via the PLC user program, be configured specially for this. Preactivation in the part program is only useful for these protection zones.

Automatic deactivation via machine data parameterization

When executing the **Geometry axis change** and **Transformation change** functions, the active protection zones can be deactivated automatically. The setting is performed NC-specifically via the machine data:

MD10618 $MN_PROTAREA_GEOAX_CHANGE_MODE

Bit	Value	Meaning
0	0	The active protection zones are deactivated during the transformation change.
	1	The active protection zones remain active during the transformation change.
1	0	The active protection zones are deactivated during the geometry axis change.
	1	The active protection zones remain active during the geometry axis change.

Display protection zone violation

Violations of activated protection zones or possible violations of preactivated protection zones, if they would be activated, are displayed via the following NC/PLC interface signals:

- DB21, ... DBX276.0 - 277.1 == 1 (machine-related protection zone 1 - 10 violated)
- DB21, ... DBX278.0 - 279.1 == 1 (channel-specific protection zone 1 - 10 violated)

Behavior in special system states

Block search with calculation

For block search with calculation, the last programmed activation state of a protection zone is always taken into account.

Program test

In the AUTOMATIC and MDI modes, activated and preactivated protection zones are also monitored in the "Program test" state.

NC RESET and end of program

The activation status of a protection zone is retained even after an `NC RESET` and end of program.

Memory requirements

The memory required for protection zones is parameterized via the following machine data:

- Persistent memory
  - MD18190 $MN_MM_NUM_PROTECT_AREA_NCK (number of available machine-defined protection zones)
  - MD28200 $MC_MM_NUM_PROTECT_AREA_CHAN (number of available channel-defined protection zones)
- Dynamic memory
  - MD28210 $MC_MM_NUM_PROTECT_AREA_ACTIVE (maximum number of protection zones that can be activated simultaneously in the channel)
  - MD28212 $MC_MM_NUM_PROTECT_AREA_CONTUR (maximum number of definable contour elements per protection zone)

See also

Definition via part program instruction (Page 128)

Definition as per system variable (Page 132)

2.3.6 Protection zone violation and temporary enabling of individual protection zones

Temporary enabling of protection zones

If a protection zone violation occurs when starting or during a traversing motion, under certain circumstances, the protection zone can be enabled, i.e. for temporary traversing. In AUTOMATIC and MDA mode as well as in JOG mode, the temporary enabling of protection zones is performed via operator actions.

A temporary enable is **only** possible for **workpiece**-related protection zones.

Tool-related protection zones must either be deactivated in the part program or via the NC/PLC interface in the "Preactivated" state.

Terminating the temporary enabling

Temporary enabling of a protection zone is terminated after the following events:

- NC RESET
- AUTOMATIC or MDA mode: The end of the block is outside the protection zone
- JOG mode: The end of the traversing motion is outside the protection zone
- The protection zone is activated

Behavior in the AUTOMATIC and MDA modes

In AUTOMATIC and MDA mode, no traversing motion is enabled into or through active protection zones:

- A traversing motion that would lead from outside into an active protection zone, is stopped at the end of the last block located outside the protection zone.
- A traversing motion that begins within an active protection zone is not started.

Temporary enabling of protection zones

If a traversing motion is stopped in AUTOMATIC or MDA mode because of a protection zone violation, this is indicated to the operator by an alarm. If the operator decides that the traversing motion can be continued, the crossing of protection zones can be enabled.

The enabling is only temporarily and is performed by triggering NC START:

DB21, ... DBX7.1 = 1 (NC START)

An alarm is displayed for each violated protection zone. An NC START signal must be triggered by the operator for each protection zone to be enabled.

The traversing motion is continued when all protection zones that have resulted in the stopping of the traversing motion, have been enabled.

Continuation of a traversing motion without temporary enabling

A traversing motion was stopped with an alarm because of a protection zone violation. If the relevant protection zone is set to the "Preactivated" state via the NC/PLC interface, the traversing motion can be continued with NC START, without the protection zone being temporarily enabled.

Increased protection against the enabling of protection zones

If the enabling of a protection zone is to be protected better than just by NC START, this must be performed by the machine manufacturer or user in the PLC user program.

Behavior in JOG mode

Simultaneous traversing of several geometry axes

In JOG mode, traversing motions can be performed simultaneously in several geometry axes. The traversing range of every participating geometry axis is limited axis-specifically at the start of the traversing motion with regard to the traversing range limits (software limit switches, working area limitation, etc.) and active protection zones. However, safe monitoring of all active protection zones cannot be guaranteed. The following is provided as feedback to the user:

- Alarm: "10704 Protection zone monitoring is not guaranteed"
- DB31, ... DBX39.0 = 1 (protection zone monitoring not guaranteed)

After the end of the traversing motion, the alarm is cleared automatically. If the current position is within an activated or preactivated protection zone, the following is performed:

- Alarm 10702 or 10703 is displayed
- Further traversing motions are disabled
- The NC/PLC interface signal for the relevant protection zone is set
  DB21, … DBX276.0 - 277.1 or DBX278.0 - 279.1 (protection zone violated)

To continue, see the paragraph below "Temporary enabling of protection zones" and Section "Activating and deactivating protection zones (Page 134)".

Example: Three protection zones activated and simultaneous traversing of two geometry axes

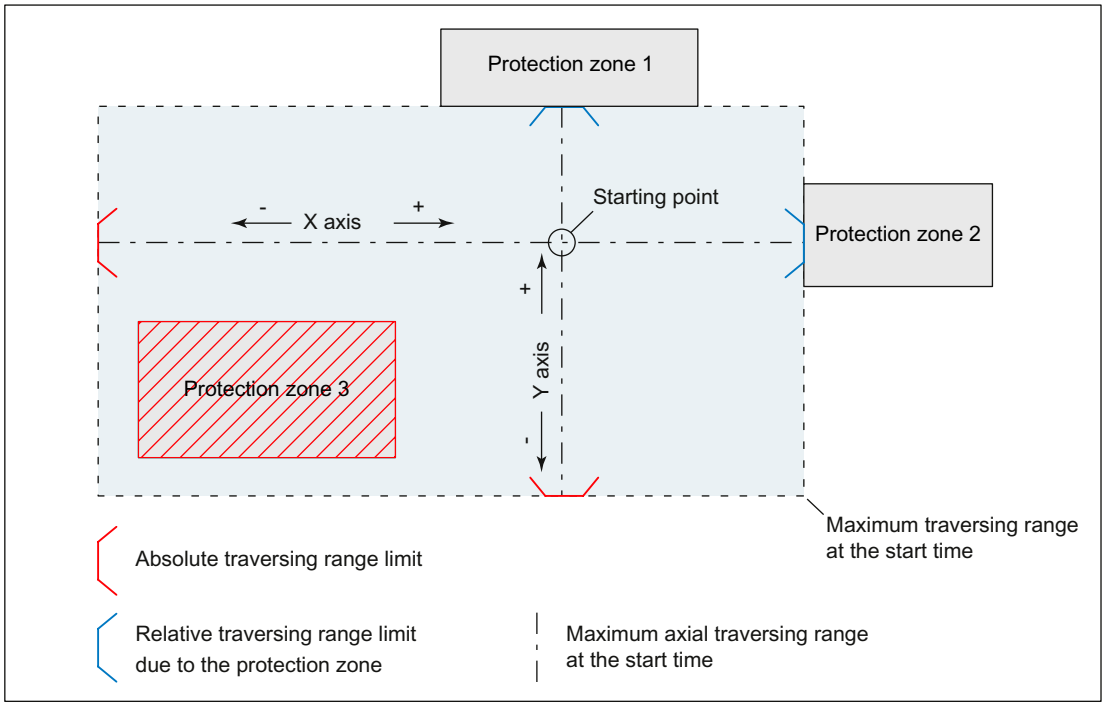


Figure 2-14 Motion range of the geometry axes at the start time

At the start time of the traversing motions of the axes X and Y, the axial traversing range limits are determined from the start time:

- X axis
  - Positive traversing direction: Protection zone 2
  - Negative traversing direction: Absolute traversing range limit, e.g. software limit switch
- Y axis
  - Positive traversing direction: Protection zone 1
  - Negative traversing direction: Absolute traversing range limit, e.g. working area limitation

The resulting maximum traversing range at the start time does not take protection zone 3 into account. Therefore, a protection zone violation in protection zone 3 is possible.

Note

Activated and preactivated protection zones are also monitored in the manual operating modes JOG, INC and DRF.

Limitation of traversing motion of an axis

If the traversing motion of an axis is limited because it has reached a protection zone, then a selfresetting alarm "Protection zone reached in JOG" is generated. The alarm text specifies the violated protection zone and the relevant axis. It is assured that no protection zone will be violated when an axis is traversing in JOG. (This response is analogous to approaching software limit switches or a working area limitation.)

The alarm is reset:

- When an axis is traversed along a path that does not lead into the protection zone
- When the protection zone is enabled
- On NC RESET

If an axis starts to move towards a protection zone when it is at a protection zone limit, then a selfresetting alarm "Protection zone reached in JOG" is output and the motion is not started.

Temporary enabling of protection zones

If a traversing motion is started within or on the limit of an activated protection zone, alarm 10702 or 10703 is displayed and the traversing motion is not started. The traversing motion can be performed when the relevant protection zone is temporarily enabled. The following actions must be performed for this:

- Create a positive edge at the NC/PLC interface signal:
  DB21, ... DBX1.1 (enable protection zone)
- Start the same traversing motion again

Note

The NC/PLC interface signal "Protection zone violated" is still set when the temporarily enabled protection zone is traversed:

DB21, … DBX276.0 – 277.1 or DBX278.0 – 279.1

The enable is reset if a motion is started that does not lead into the enabled protection zone.

If further protection zones are affected by the traversing motion, additional alarms 10702 or 10703 are displayed for each protection zone. The protection zones displayed in the alarms can be enabled by creating further positive edges on the NC/PLC interface signal:

DB21, ... DBX1.1 (enable protection zone)

Behavior at change of operating mode

The protection zones temporarily enabled in JOG mode are retained after a change to AUTOMATIC or MDA mode. The temporary enables set the in the AUTOMATIC or MDA mode are also retained after a change to JOG mode.

Reset of an enable

The enable is reset internally and on the NC/PLC interface at the next standstill of a geometry axis for which the temporarily enabled protection zone has been completely exited:

DB21, … DBX276.0 - 277.1 or DBX278.0 - 279.1 = 0 (protection zone violated)

Behavior with axis replacement

With regard to the protection zones after an axis replacement, the starting position is the last position approached in the relinquishing channel. Traversing motions in the channel taking over are not taken into account. For this reason, you must make sure that an axis replacement is not performed from a position with protection zone violation.

Behavior for superimposed motions

Superimposed motions that are included in the main run, cannot be taken into account by the block preparation with regard to the active protection zones.

This results in the following reactions:

- Alarm: "10704 Protection zone monitoring is not guaranteed"
- DB31, ... DBX39.0 = 1 (protection zone monitoring not guaranteed)

2.3.7 Restrictions in protection zones

Restrictions in protection-zone monitoring

No protection-zone monitoring is possible under the following conditions:

- Orientation axes
- Protection-zone monitoring for fixed machine-related protection zones with TRANSMIT or peripheral surface transformation.

  Exception: Protection zones defined with rotation symmetry around the spindle axis. Here, no DRF offset must be active.
- Mutual monitoring of tool-related protection zones

Positioning axes

For positioning axes, only the programmed block end point is monitored.

An alarm is displayed during the traversing motion of the positioning axes:

Alarm: "10704 Protection-zone monitoring is not guaranteed".

Axis interchange

If an axis intended for the axis interchange is not active in a channel, the position of the axis last approached in the channel is taken as the current position. If this axis has not yet been traversed in the channel, zero is taken as the position.

Machine-related protection zones

A machine-related protection zone or its contour is defined using the geometry axis, i.e. with reference to the basic coordinate system (BCS) of a channel. In order that correct protection-zone monitoring can take place in all channels in which the machine-related protection zone is active, the basic coordinate system (BCS) of all affected channels must be identical (position of the coordinate point of origin with respect to the machine zero point and orientation of the coordinate axes).

2.4 Supplementary conditions

2.4.1 Axis monitoring functions

Settings

For correct operation of the monitoring, the following settings must be made or checked, in addition to the machine data mentioned:

General

- MD31030 $MA_LEADSCREW_PITCH (leadscrew pitch)
- MD31050 $MA_DRIVE_AX_RATIO_DENOM (denominator load gearbox)
- MD31060 $MA_DRIVE_AX_RATIO_NUMERA (numerator load gearbox)
- MD31070 $MA_DRIVE_ENC_RATIO_DENOM (denominator measuring gearbox)
- MD31080 $MA_DRIVE_ENC_RATIO_NUMERA (numerator measuring gearbox)
- MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)
- Encoder resolution

  For the appropriate machine data, see Section "G2: Velocities, setpoint / actual value systems, closed-loop control (Page 331)".

Only drives with analog speed setpoint interface

- MD32260 $MA_RATED_VELO (rated motor speed)
- MD32250 $MA_RATED_OUTVAL (rated output voltage)

2.5 Examples

2.5.1 Axis monitoring functions

2.5.1.1 Working area limitation in WCS/SZS

Available channel axes

Four axes are defined in the channel: X, Y, Z and A

The A-axis is a rotary axis (not modulo).

Parameterize the number of working area limitation groups

Three working area limitation groups will be provided:

MD28600 $MC_MM_NUM_WORKAREA_CS_GROUP = 3

Define the working area limitation groups

Additionally two working area limitation groups will be defined:

Working area limitation group 1

In the first working area limitation group the axes in the SZS coordinate system will be limited:

- X axis in the plus direction: 10 mm
- X axis in the minus direction: No limitation
- Y axis in the plus direction: No limitation
- Y axis in the minus direction: 25 mm
- Z axis in the plus direction: No limitation
- Z axis in the minus direction: No limitation
- A axis in the plus direction: 10 degrees
- A axis in the minus direction: -40 degrees

The system variables are assigned as follows:

Program code	Comment
N1 $P_WORKAREA_CS_COORD_SYSTEM[1]=3	; The working area limitation of the working area limitation group 1 applies in the SZS.
N10 $P_WORKAREA_CS_PLUS_ENABLE[1,X]=TRUE	
N11 $P_WORKAREA_CS_LIMIT_PLUS[1,X]=10	
N12 $P_WORKAREA_CS_MINUS_ENABLE[1,X]=FALSE	
N20 $P_WORKAREA_CS_PLUS_ENABLE[1,Y]=FALSE	
N22 $P_WORKAREA_CS_MINUS_ENABLE[1,Y]=TRUE	
N23 $P_WORKAREA_CS_LIMIT_MINUS[1,Y]=25	
N30 $P_WORKAREA_CS_PLUS_ENABLE[1,Z]=FALSE	
N32 $P_WORKAREA_CS_MINUS_ENABLE[1,Z]=FALSE	
N40 $P_WORKAREA_CS_PLUS_ENABLE[1,A]=TRUE	
N41 $P_WORKAREA_CS_LIMIT_PLUS[1,A]=10	
N42 $P_WORKAREA_CS_MINUS_ENABLE[1,A]=TRUE	
N43 $P_WORKAREA_CS_LIMIT_MINUS[1,A]=-40	

Working area limitation group 2

In the second working area limitation group the axes in the WCS coordinate system can be limited:

- X axis in the plus direction: 10 mm
- X axis in the minus direction: No limitation
- Y axis in the plus direction: 34 mm
- Y axis in the minus direction: -25 mm
- Z axis in the plus direction: No limitation
- Z axis in the minus direction: -600 mm
- A axis in the plus direction: No limitation
- A axis in the minus direction: No limitation

The system variables are assigned as follows:

Program code	Comment
N51 $P_WORKAREA_CS_COORD_SYSTEM[2]=1	; The working area limitation of working area limitation group 2 applies in the WCS.
N60 $P_WORKAREA_CS_PLUS_ENABLE[2,X]=TRUE	
N61 $P_WORKAREA_CS_LIMIT_PLUS[2,X]=10	
N62 $P_WORKAREA_CS_MINUS_ENABLE[2,X]=FALSE	
N70 $P_WORKAREA_CS_PLUS_ENABLE[2,Y]=TRUE	
N73 $P_WORKAREA_CS_LIMIT_PLUS[2,Y]=34	
N72 $P_WORKAREA_CS_MINUS_ENABLE[2,Y]=TRUE	
N73 $P_WORKAREA_CS_LIMIT_MINUS[2,Y]=-25	
N80 $P_WORKAREA_CS_PLUS_ENABLE[2,Z]=FALSE	
N82 $P_WORKAREA_CS_MINUS_ENABLE[2,Z]=TRUE	
N83 $P_WORKAREA_CS_LIMIT_PLUS[2,Z]=-600	
N90 $P_WORKAREA_CS_PLUS_ENABLE[2,A]=FALSE	
N92 $P_WORKAREA_CS_MINUS_ENABLE[2,A]=FALSE	

Activate working area limitation group 2

In order to activate the working area limitation group 2, the following instruction must exist in the part program:

```
...
N100 WALCS2 ...
...
```

2.5.2 Protection zones

2.5.2.1 Definition and activation of protection zones

Requirement

The following internal protection zones are to be defined for a turning machine:

- One machine- and workpiece-related protection zone for the spindle chuck, without limitation in the third dimension
- One channel-specific protection zone for the workpiece, without limitation in the third dimension
- One channel-specific, tool-related protection zone for the toolholder, without limitation in the third dimension

The workpiece zero is placed on the machine zero to define the protection zone for the workpiece.

When activated, the protection zone is then offset by 100 mm in the Z axis in the positive direction.

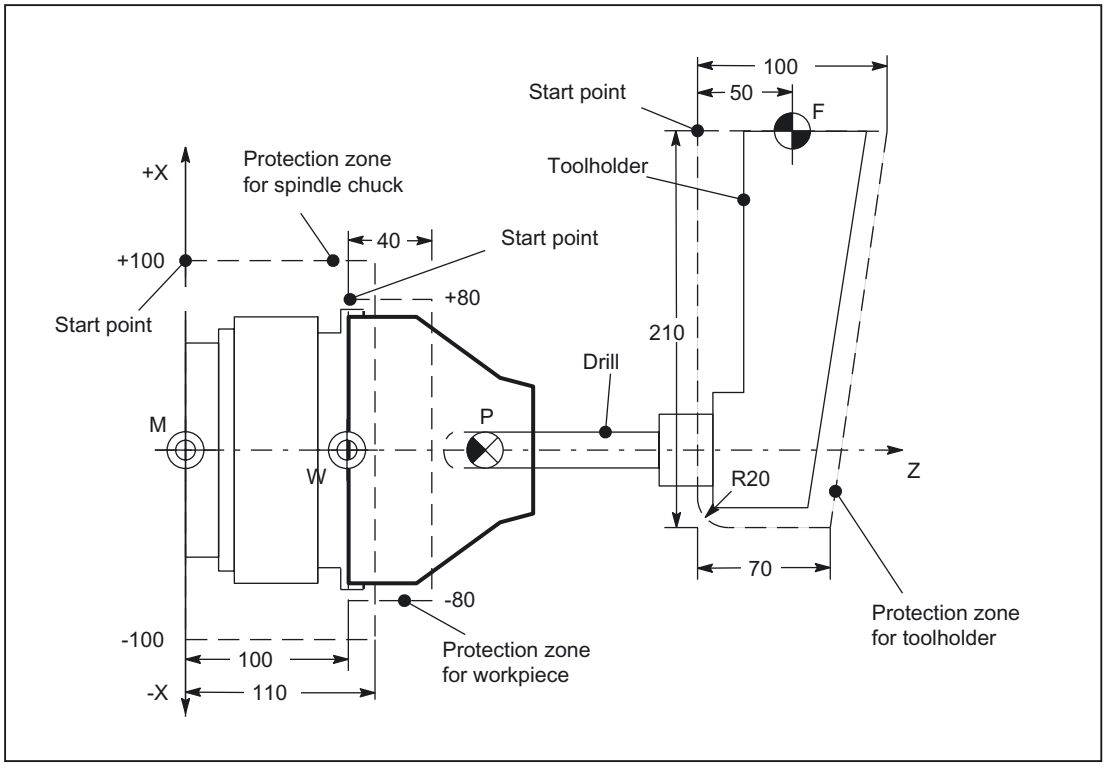


Figure 2-15 Example of protection zones on a turning machine

Protection zone definition in the part program

Table 2- 1 Part program excerpt for protection zone definition:

Program code	Comment
DEF INT AB	
G18	; Definition of the working plane
NPROTDEF(1,FALSE,0,0,0)	; Start of definition: Protection zone for spindle chuck
G01 X100 Z0	; Contour definition: 1st contour element
G01 X-100 Z0	; Contour definition: 2nd contour element
G01 X-100 Z110	; Contour definition: 3rd contour element
G01 X100 Z110	; Contour definition: 4th contour element
G01 X100 Z0	; Contour definition: 5th contour element
EXECUTE(AB)	; End of definition: Protection zone for spindle chuck
CPROTDEF(1,FALSE,0,0,0)	; Start of definition: Protection zone for workpiece
G01 X80 Z0	; Contour definition: 1st contour element
G01 X-80 Z0	; Contour definition: 2nd contour element
G01 X-80 Z40	; Contour definition: 3rd contour element
G01 X80 Z40	; Contour definition: 4th contour element
G01 X80 Z0	; Contour definition: 5th contour element
EXECUTE(AB)	; End of definition: Protection zone for workpiece
CPROTDEF(2,TRUE,0,0,0)	; Start of definition: Protection zone for toolholder
G01 X0 Z-50	; Contour definition: 1st contour element
G01 X-190 Z-50	; Contour definition: 2nd contour element
G03 X-210 Z-30 I-20	; Contour definition: 3rd contour element
G01 X-210 Z20	; Contour definition: 4th contour element
G01 X0 Z50	; Contour definition: 5th contour element
G01 X0 Z-50	; Contour definition: 6th contour element
EXECUTE(AB)	; End of definition: Protection zone for toolholder

Protection zone definition with system variables

Table 2- 2 Protection zone: Spindle chuck

System variable	Value	Remark
$SN_PA_ACTIV_IMMED[0]	0	; Protection zone for spindle chuck not immediately active
$SN_PA_T_W[0]	" "	; Machine-related protection zone for spindle chuck
$SN_PA_ORI[0]	1	; Orientation of the protection zone: 1= 3rd and 1st geometry axis
$SN_PA_LIM_3DIM[0]	0	; Type of limitation in the third dimension: 0 = No limit
$SN_PA_PLUS_LIM[0]	0	; Value of the limit in the positive direction in the 3rd dimension
$SN_PA_MINUS_LIM[0]	0	; Value of the limitation in the negative direction in the 3rd dimension
$SN_PA_CONT_NUM[0]	4	; Number of valid contour elements
$SN_PA_CONT_TYP[0,0]	1	; Contour type[i] : 1 = G1 for even, ; Protection zone for spindle chuck, contour element 0
$SN_PA_CONT_TYP[0,1]	1	; Contour type[i] : 1 = G1 for even, ; Protection zone for spindle chuck, contour element 1
$SN_PA_CONT_TYP[0,2]	1	; Contour type[i] : 1 = G1 for even, ; Protection zone for spindle chuck, contour element 2
$SN_PA_CONT_TYP[0,3]	1	; Contour type[i] : 1 = G1 for even, ; Protection zone for spindle chuck, contour element 3
$SN_PA_CONT_TYP[0,4]	0	; Contour type[i] : 0 = not defined, ; Protection zone for spindle chuck, contour element 4
$SN_PA_CONT_TYP[0,5]	0	; Contour type[i] : 0 = not defined, ; Protection zone for spindle chuck, contour element 5
$SN_PA_CONT_TYP[0,6]	0	; Contour type[i] : 0 = not defined, ; Protection zone for spindle chuck, contour element 6
$SN_PA_CONT_TYP[0,7]	0	; Contour type[i] : 0 = not defined, ; Protection zone for spindle chuck, contour element 7
$SN_PA_CONT_TYP[0,8]	0	; Contour type[i] : 0 = not defined, ; Protection zone for spindle chuck, contour element 8
$SN_PA_CONT_TYP[0,9]	0	; Contour type[i] : 0 = not defined, ; Protection zone for spindle chuck, contour element 9
$SN_PA_CONT_ORD[0,0]	-100	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 0
$SN_PA_CONT_ORD[0,1]	-100	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 1
$SN_PA_CONT_ORD[0,2]	100	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 2
$SN_PA_CONT_ORD[0,3]	100	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 3
$SN_PA_CONT_ORD[0,4]	0	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 4
$SN_PA_CONT_ORD[0,5]	0	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 5

$SN_PA_CONT_ORD[0,6]	0	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 6
$SN_PA_CONT_ORD[0,7]	0	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 7
$SN_PA_CONT_ORD[0,8]	0	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 8
$SN_PA_CONT_ORD[0,9]	0	; Endpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 9
$SN_PA_CONT_ABS[0,0]	0	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 0
$SN_PA_CONT_ABS[0,1]	110	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 1
$SN_PA_CONT_ABS[0,2]	110	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 2
$SN_PA_CONT_ABS[0,3]	0	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 3
$SN_PA_CONT_ABS[0,4]	0	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 4
$SN_PA_CONT_ABS[0,5]	0	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 5
$SN_PA_CONT_ABS[0,6]	0	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 6
$SN_PA_CONT_ABS[0,7]	0	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 7
$SN_PA_CONT_ABS[0,8]	0	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 8
$SN_PA_CONT_ABS[0,9]	0	; Endpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 9
$SN_PA_CENT_ORD[0,0]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 0
$SN_PA_CENT_ORD[0.1]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 1
$SN_PA_CENT_ORD[0,2]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 2
$SN_PA_CENT_ORD[0,3]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 3
$SN_PA_CENT_ORD[0,4]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 4
$SN_PA_CENT_ORD[0,5]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 5
$SN_PA_CENT_ORD[0,6]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 6
$SN_PA_CENT_ORD[0,7]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 7
$SN_PA_CENT_ORD[0,8]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 8
$SN_PA_CENT_ORD[0,9]	0	; Midpoint of contour[i], ordinate value ; Protection zone for spindle chuck, contour element 9

$SN_PA_CENT_ABS[0,0]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 0
$SN_PA_CENT_ABS[0.1]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 1
$SN_PA_CENT_ABS[0,2]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 2
$SN_PA_CENT_ABS[0,3]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 3
$SN_PA_CENT_ABS[0,4]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 4
$SN_PA_CENT_ABS[0,5]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 5
$SN_PA_CENT_ABS[0,6]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 6
$SN_PA_CENT_ABS[0,7]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 7
$SN_PA_CENT_ABS[0,8]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 8
$SN_PA_CENT_ABS[0,9]	0	; Midpoint of contour[i], abscissa value ; Protection zone for spindle chuck, contour element 9

Table 2- 3 Protection zone: Workpiece and toolholder

System variable	Value	Remark
$SN_PA_ACTIV_IMMED[0]	0	; Protection zone for workpiece not immediately active
$SN_PA_ACTIV_IMMED[1]	0	; Protection zone for toolholder not immediately active
$SC_PA_TW[0]	" "	; Protection zone for workpiece, channel-specific
$SC_PA_TW[1]	'H01'	; Protection zone for toolholder, channel-specific
$SC_PA_ORI[0]	1	; Orientation of the protection zone: 1= 3rd and 1st geometry axis ; Protection zone for workpiece
$SC_PA_ORI[1]	1	; Orientation of the protection zone: 1= 3rd and 1st geometry axis ; Protection zone for toolholder
$SC_PA_LIM_3DIM[0]	0	; Type of limitation in the third dimension: 0 = no limitation ; Protection zone for workpiece toolholder 0
$SC_PA_LIM_3DIM[1]	0	; Type of limitation in the third dimension: 0 = no limitation ; Protection zone for toolholder
$SC_PA_PLUS_LIM[0]	0	; Value of limitation in positive direction in the third dimension ; Protection zone for workpiece
$SC_PA_PLUS_LIM[1]	0	; Value of limitation in positive direction in the third dimension ; Protection zone for toolholder
$SC_PA_MINUS_LIM[0]	0	; Value of limitation in negative direction in the third dimension ; Protection zone for workpiece
$SC_PA_MINUS_LIM[1]	0	; Value of limitation in negative direction in the third dimension ; Protection zone for toolholder

$SC_PA_CONT_NUM[0]	4	; Number of valid contour elements ; Protection zone for workpiece
$SC_PA_CONT_NUM[1]	5	; Number of valid contour elements ; Protection zone for toolholder 1
$SN_PA_CONT_TYP[0,0]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for workpiece, contour element 0
$SN_PA_CONT_TYP[0,1]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for workpiece, contour element 1
$SN_PA_CONT_TYP[0,2]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for workpiece, contour element 2
$SN_PA_CONT_TYP[0,3]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for workpiece, contour element 3
$SN_PA_CONT_TYP[0,4]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for workpiece, contour element 4
$SN_PA_CONT_TYP[0,5]	0	; Contour type[i] : 0 = not defined ; Protection zone for workpiece, contour element 5
$SN_PA_CONT_TYP[0.6]	0	; Contour type[i] : 0 = not defined ; Protection zone for workpiece, contour element 6
$SN_PA_CONT_TYP[0,7]	0	; Contour type[i] : 0 = not defined ; Protection zone for workpiece, contour element 7
$SN_PA_CONT_TYP[0,8]	0	; Contour type[i] : 0 = not defined ; Protection zone for workpiece, contour element 8
$SN_PA_CONT_TYP[0,9]	0	; Contour type[i] : 0 = not defined ; Protection zone for workpiece, contour element 9
$SN_PA_CONT_TYP[1,0]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for toolholder, contour element 0
$SN_PA_CONT_TYP[1,1]	3	; Contour type[i] : 3 = G3 for circuit element, counter-clockwise ; Protection zone for toolholder, contour element 1
$SN_PA_CONT_TYP[1,2]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for toolholder, contour element 2
$SN_PA_CONT_TYP[1,3]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for toolholder, contour element 3
$SN_PA_CONT_TYP[1,4]	1	; Contour type[i] : 1 = G1 for even ; Protection zone for toolholder, contour element 4
$SN_PA_CONT_TYP[1,5]	0	; Contour type[i] : 0 = not defined ; Protection zone for toolholder, contour element 5
$SN_PA_CONT_TYP[1,6]	0	; Contour type[i] : 0 = not defined ; Protection zone for toolholder, contour element 6
$SN_PA_CONT_TYP[1,7]	0	; Contour type[i] : 0 = not defined ; Protection zone for toolholder, contour element 7
$SN_PA_CONT_TYP[1,8]	0	; Contour type[i] : 0 = not defined ; Protection zone for toolholder, contour element 8
$SN_PA_CONT_TYP[1,9]	0	; Contour type[i] : 0 = not defined ; Protection zone for toolholder, contour element 9
$SN_PA_CONT_ORD[0,0]	-80	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 0
$SN_PA_CONT_ORD[0,1]	-80	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 1

$SN_PA_CONT_ORD[0,2]	80	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 2
$SN_PA_CONT_ORD[0,3]	80	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 3
$SN_PA_CONT_ORD[0,4]	0	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 4
$SN_PA_CONT_ORD[0,5]	0	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 5
$SN_PA_CONT_ORD[0,6]	0	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 6
$SN_PA_CONT_ORD[0,7]	0	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 7
$SN_PA_CONT_ORD[0,8]	0	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 8
$SN_PA_CONT_ORD[0,9]	0	; Endpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 9
$SN_PA_CONT_ORD[1,0]	-190	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 0
$SN_PA_CONT_ORD[1,1]	-210	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 1
$SN_PA_CONT_ORD[1,2]	-210	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 2
$SN_PA_CONT_ORD[1,3]	0	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 3
$SN_PA_CONT_ORD[1,4]	0	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 4
$SN_PA_CONT_ORD[1,5]	0	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 5
$SN_PA_CONT_ORD[1,6]	0	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 6
$SN_PA_CONT_ORD[1,7]	0	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 7
$SN_PA_CONT_ORD[1,8]	0	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 8
$SN_PA_CONT_ORD[1,9]	0	; Endpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 9
$SN_PA_CONT_ABS[0,0]	0	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 0
$SN_PA_CONT_ABS[0,1]	40	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 1
$SN_PA_CONT_ABS[0,2]	40	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 2
$SN_PA_CONT_ABS[0,3]	0	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 3
$SN_PA_CONT_ABS[0,4]	-50	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 4
$SN_PA_CONT_ABS[0,5]	0	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 5

$SN_PA_CONT_ABS[0,6]	0	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 6
$SN_PA_CONT_ABS[0,7]	0	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 7
$SN_PA_CONT_ABS[0,8]	0	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 8
$SN_PA_CONT_ABS[0,9]	0	; Endpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 9
$SN_PA_CONT_ABS[1,0]	-50	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 0
$SN_PA_CONT_ABS[1,1]	-30	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 1
$SN_PA_CONT_ABS[1,2]	20	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 2
$SN_PA_CONT_ABS[1,3]	50	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 3
$SN_PA_CONT_ABS[1,4]	-50	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 4
$SN_PA_CONT_ABS[1,5]	0	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 5
$SN_PA_CONT_ABS[1,6]	0	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 6
$SN_PA_CONT_ABS[1,7]	0	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 7
$SN_PA_CONT_ABS[1,8]	0	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 8
$SN_PA_CONT_ABS[1,9]	0	; Endpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 9
$SN_PA_CENT_ORD[0,0]	0	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 0
$SN_PA_CENT_ORD[0.1]	-190	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 1
$SN_PA_CENT_ORD[0,2]	0	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 2
$SN_PA_CENT_ORD[0,3]	0	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 3
$SN_PA_CENT_ORD[0,4]	0	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 4
$SN_PA_CENT_ORD[0,5]	0	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 5
$SN_PA_CENT_ORD[0,6]	0	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 6
$SN_PA_CENT_ORD[0,7]	0	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 7
$SN_PA_CENT_ORD[0,8]	0	; Midpoint of contour[i], ordinate value ; Protection zone for workpiece, contour element 8
$SN_PA_CENT_ORD[0,9]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 9

$SN_PA_CENT_ORD[1.0]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 0
$SN_PA_CENT_ORD[1.1]	-190	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 1
$SN_PA_CENT_ORD[1.2]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 2
$SN_PA_CENT_ORD[1.3]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 3
$SN_PA_CENT_ORD[1.4]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 4
$SN_PA_CENT_ORD[1.5]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 5
$SN_PA_CENT_ORD[1.6]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 6
$SN_PA_CENT_ORD[1.7]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 7
$SN_PA_CENT_ORD[1.8]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 8
$SN_PA_CENT_ORD[1.9]	0	; Midpoint of contour[i], ordinate value ; Protection zone for toolholder, contour element 9
$SN_PA_CENT_ABS[0,0]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 0
$SN_PA_CENT_ABS[0.1]	-30	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 1
$SN_PA_CENT_ABS[0,2]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 2
$SN_PA_CENT_ABS[0,3]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 3
$SN_PA_CENT_ABS[0,4]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 4
$SN_PA_CENT_ABS[0,5]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 5
$SN_PA_CENT_ABS[0,6]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 6
$SN_PA_CENT_ABS[0,7]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 7
$SN_PA_CENT_ABS[0,8]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 8
$SN_PA_CENT_ABS[0,9]	0	; Midpoint of contour[i], abscissa value ; Protection zone for workpiece, contour element 9
$SN_PA_CENT_ABS[1.0]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 0
$SN_PA_CENT_ABS[1.1]	-30	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 1
$SN_PA_CENT_ABS[1.2]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 2
$SN_PA_CENT_ABS[1.3]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 3

$SN_PA_CENT_ABS[1.4]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 4
$SN_PA_CENT_ABS[1.5]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 5
$SN_PA_CENT_ABS[1.6]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 6
$SN_PA_CENT_ABS[1.7]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 7
$SN_PA_CENT_ABS[1.8]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 8
$SN_PA_CENT_ABS[1.9]	0	; Midpoint of contour[i], abscissa value ; Protection zone for toolholder, contour element 9

Activation

Table 2- 4 Part program excerpt for activating the three protection zones for spindle chuck, workpiece, and toolholder:

```
Program code                          Comment
NPROT(1, 2, 0, 0, 0)                  ; Protection zone: Spindle chuck
CPROT(1, 2, 0, 0, 100)                ; Protection zone: Workpiece with 100 mm offset
                                        in the Z axis
CPROT(2, 2, 0, 0, 0)                  ; Protection zone: Toolholder
```

2.6 Data lists

2.6.1 Machine data

2.6.1.1 NC-specific machine data

Axis monitoring

Number	Identifier: $MN_	Description
10604	WALIM_GEOAX_CHANGE_MODE	Working area limitation during switchover of geometry axes
10710	PROG_SD_RESET_SAVE_TAB	Setting data to be updated

Protection zones

Number	Identifier: $MN_	Description
10618	PROTAREA_GEOAX_CHANGE_MODE	Protection zone for switchover of geo axes
18190	MM_NUM_PROTECT_AREA_NCK	Number of files for machinerelated protection zones

2.6.1.2 Channelspecific machine data

Axis monitoring functions

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Initial setting of the G groups
21020	WORKAREA_WITH_TOOL_RADIUS	Allowance for tool radius with working area limitation
24130	TRAFO_INCLUDES_TOOL_1	Tool handling with active transformation 1.
24230	TRAFO_INCLUDES_TOOL_2	Tool handling with active transformation 2.
24330	TRAFO_INCLUDES_TOOL_3	Tool handling with active transformation 3.
24426	TRAFO_INCLUDES_TOOL_4	Tool handling with active transformation 4.
24436	TRAFO_INCLUDES_TOOL_5	Tool handling with active transformation 5.
24446	TRAFO_INCLUDES_TOOL_6	Tool handling with active transformation 6.
24456	TRAFO_INCLUDES_TOOL_7	Tool handling with active transformation 7.
24466	TRAFO_INCLUDES_TOOL_8	Tool handling with active transformation 8.
24476	TRAFO_INCLUDES_TOOL_9	Tool handling with active transformation 9.
24486	TRAFO_INCLUDES_TOOL_10	Tool handling with active transformation 10.
25106	TRAFO_INCLUDES_TOOL_11	Tool handling with active transformation 11.
25116	TRAFO_INCLUDES_TOOL_12	Tool handling with active transformation 12.
25126	TRAFO_INCLUDES_TOOL_13	Tool handling with active transformation 13.
25136	TRAFO_INCLUDES_TOOL_14	Tool handling with active transformation 14.
25146	TRAFO_INCLUDES_TOOL_15	Tool handling with active transformation 15.
25156	TRAFO_INCLUDES_TOOL_16	Tool handling with active transformation 16.
25166	TRAFO_INCLUDES_TOOL_17	Tool handling with active transformation 17.
25176	TRAFO_INCLUDES_TOOL_18	Tool handling with active transformation 18.
25186	TRAFO_INCLUDES_TOOL_19	Tool handling with active transformation 19.
25196	TRAFO_INCLUDES_TOOL_20	Tool handling with active transformation 20.
28600	MM_NUM_WORKAREA_CS_GROUPS	Number of coordinate system-specific working area limitations

Protection zones

Number	Identifier: $MC_	Description
28200	MM_NUM_PROTECT_AREA_CHAN (SRAM)	Number of files for channelspecific protection zones
28210	MM_NUM_PROTECT_AREA_ACTIVE	Number of simultaneously active protection zones in one channel
28212	MM_NUM_PROTECT_AREA_CONTUR	Elements for active protection zones (DRAM)

2.6.1.3 Axis/spindlespecific machine data

Axis monitoring functions

Number	Identifier: $MA_	Description
30240	ENC_TYPE	Encoder type of the actual value sensing (position actual value)
30310	ROT_IS_MODULO	Modulo conversion for rotary axis / spindle
30800	WORK_AREA_CHECK_TYPE	Type of checking of working area limits
32200	POSCTRL_GAIN [n]	Servo gain factor
32250	RATED_OUTVAL	Rated output voltage
32260	RATED_VELO	Rated motor speed
32300	MAX_AX_ACCEL	Maximum axis acceleration
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control loop for feedforward control
32910	DYN_MATCH_TIME [n]	Time constant for dynamic response adaptation
35160	SPIND_EXTERN_VELO_LIMIT	Spindle speed limitation via PLCC
36000	STOP_LIMIT_COARSE	Exact stop coarse
36010	STOP_LIMIT_FINE	Exact stop fine
36020	POSITIONING_TIME	Delay time exact stop fine
36030	STANDSTILL_POS_TOL	Zero speed tolerance
36040	STANDSTILL_DELAY_TIME	Delay time zero-speed monitoring
36050	CLAMP_POS_TOL	Clamping tolerance with IS "Clamping active"
36052	STOP_ON_CLAMPING	Special functions for clamped axis
36060	STANDSTILL_VELO_TOL	Maximum velocity/speed "Axis/spindle stationary"
36100	POS_LIMIT_MINUS	1st software limit switch minus
36110	POS_LIMIT_PLUS	1st software limit switch plus
36120	POS_LIMIT_MINUS2	2nd software limit switch minus
36130	POS_LIMIT_PLUS2	2nd software limit switch plus
36200	AX_VELO_LIMIT	Threshold value for velocity monitoring
36210	CTRLOUT_LIMIT	Maximum speed setpoint
36220	CTRLOUT_LIMIT_TIME	Delay time for speed-setpoint monitoring
36300	ENC_FREQ_LIMIT	Encoder limit frequency
36302	ENC_FREQ_LIMIT_LOW	Encoder limit frequency for encoder resynchronization
36310	ENC_ZERO_MONITORING	Zero-mark monitoring
36312	ENC_ABS_ZEROMON_WARNING	Zero-mark monitoring warning threshold
36400	CONTOUR_TOL	Tolerance band contour monitoring
36500	ENC_CHANGE_TOL	Maximum tolerance for position actual value switchover
36510	ENC_DIFF_TOL	Measuring system synchronism tolerance
36600	BRAKE_MODE_CHOICE	Deceleration behavior on hardware limit switch
36610	AX_EMERGENCY_STOP_TIME	Maximum duration of the braking ramp for faults
36620	SERVO_DISABLE_DELAY_TIME	Cutout delay controller enable

Protection zones

None

2.6.2 Setting data

2.6.2.1 Axis/spindlespecific setting data

Axis monitoring functions

Number	Identifier: $SA_	Description
43400	WORKAREA_PLUS_ENABLE	Working-area limitation active in positive direction
43410	WORKAREA_MINUS_ENABLE	Working-area limitation active in negative direction
43420	WORKAREA_LIMIT_PLUS	Working-area limitation plus
43430	WORKAREA_LIMIT_MINUS	Working-area limitation minus

Protection zones

None

2.6.3 Signals

2.6.3.1 Signals to channel

Axis monitoring functions

None

Protection zones

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Enable protection zones	DB21, ...DBX1.1	DB3200.DBX1.1
Feed disable	DB21, ...DBX6.0	DB3200.DBX6.0
Activate machinerelated protection zones 1-8	DB21, ...DBX8.0-7	DB3200.DBX8.0-7
Activate machinerelated protection zone 9	DB21, ...DBX9.0	DB3200.DBX9.0
Activate machinerelated protection zone 10	DB21, ...DBX9.1	DB3200.DBX9.1
Activate channelspecific protection zone 1-8	DB21, ...DBX10.0-7	DB3200.DBX10.0-7
Activate channelspecific protection zone 9	DB21, ...DBX11.0	DB3200.DBX11.0
Activate channelspecific protection zone 10	DB21, ...DBX11.1	DB3200.DBX11.1

2.6.3.2 Signals from channel

Axis monitoring functions

None

Protection zones

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Machinerelated protection zones 1-8 preactivated	DB21, ...DBX272.0-7	DB3300.DBX8.0-7
Machinerelated protection zone 9 preactivated	DB21, ...DBX273.0	DB3300.DBX9.0
Machinerelated protection zone 10 preactivated	DB21, ...DBX273.1	DB3300.DBX9.1
Channelspecific protection zones 1-8 preactivated	DB21, ...DBX274.0-7	DB3300.DBX10.0-7
Channelspecific protection zone 9 preactivated	DB21, ...DBX275.0	DB3300.DBX11.0
Channelspecific protection zone 10 preactivated	DB21, ...DBX275.1	DB3300.DBX11.1
Machinerelated protection zones 1-8 violated	DB21, ...DBX276.0-7	DB3300.DBX12.0-7
Machinerelated protection zone 9 violated	DB21, ...DBX277.0	DB3300.DBX13.0
Machinerelated protection zone 10 violated	DB21, ...DBX277.1	DB3300.DBX13.1
Channelspecific protection zones 1-8 violated	DB21, ...DBX278.0-7	DB3300.DBX14.0-7
Channelspecific protection zone 9 violated	DB21, ...DBX279.0	DB3300.DBX15.0
Channelspecific protection zone 10 violated	DB21, ...DBX279.1	DB3300.DBX15.1

2.6.3.3 Signals to axis/spindle

Axis monitoring functions

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Follow-up mode	DB31, ...DBX1.4	DB380x.DBX1.4
Position measuring system 1 / 2	DB31, ...DBX1.5/6	DB380x.DBX1.5/6
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Clamping in progress	DB31, ...DBX2.3	DB380x.DBX2.3
Velocity/spindle speed limitation	DB31, ...DBX3.6	DB380x.DBX3.6
Feed stop/spindle stop	DB31, ...DBX4.3	DB380x.DBX4.3
Hardware limit switch minus/Hardware limit switch plus	DB31, ...DBX12.0/1	DB380x.DBX1000.0/1
Software limit switch minus / 2nd software limit switch plus	DB31, ...DBX12.2/3	DB380x.DBX1000.2/3
Encoder limit frequency exceeded 1 / 2	DB31, ...DBX60.2/3	DB390x.DBX0.2
Referenced/synchronized 1 / 2	DB31, ...DBX60.4/5	DB390x.DBX0.4/5
Traverse command minus / plus	DB31, ...DBX64.6/7	DB390x.DBX4.6/7

Protection zones

None

B1: Continuous-path mode, Exact stop, Look Ahead 3

3.1 Brief Description

Exact stop or exact stop mode

In exact stop traversing mode, all axes involved in the traversing motion (except axes of modal traversing modes) are decelerated at the end of each block until they come to a standstill. The transition to the next block occurs only when all axes involved in the traversing motion have reached their programmed target position with subject to the selected exact stop criterion.

Continuous-path mode

In continuous-path mode, the NC attempts to keep the programmed path velocity as constant as possible. In particular, deceleration of the path axes at the block limits of the part program is to be avoided.

LookAhead

LookAhead is a function for optimizing the continuous path mode.

Smooth machining of workpieces is necessary to ensure a high-quality surface finish. For this reason, path velocity variations should be avoided during machining whenever possible. Without LookAhead, the NC only takes the traversing block immediately following the current traversing block into consideration when determining the possible path velocity. If the following block contains only a short path, the NC must reduce the path velocity (decelerate in the current block) to be able to stop in time at the end of the next block, if necessary.

When the NC "looks ahead" over a configurable number of traversing blocks following the current traversing block, a much higher path velocity can be attained under certain circumstances because the NC now has considerably more traversing blocks and more path available for calculation.

This results in the following advantages:

- Machining with higher path velocities on average
- Improved surface quality by avoiding deceleration and acceleration processes

Smoothing the path velocity

"Smoothing the path velocity" is a function especially for applications (such as high speed milling in mold and die production) that require an extremely steady path velocity. Deceleration and acceleration processes that would cause high-frequency excitations of machine resonances are avoided with the "Smoothing the path velocity" function.

This results in the following advantages:

- Improved surface quality and machining time by avoiding excitation of machine resonances.
- Constant profile of path velocity and cutting rates by avoiding "unnecessary" acceleration processes, i.e. acceleration processes that do not greatly improve the program run time.

Adaptation of the dynamic path response

In addition to "smoothing the path velocity", "dynamic path response adaptation" is another function for avoiding high-frequency excitations of machine resonances while optimizing the dynamic path response at the same time. To this end, highly frequent changes in path velocity are automatically executed with lower jerk or acceleration values than the dynamic response limit value parameters assigned in the machine data.

Thus, with low-frequency changes in path velocity, the full dynamic response limit values apply, whereas with high-frequency changes, only the reduced dynamic response limit values act due to the automatic dynamic response adaptation.

Dynamic response mode for path interpolation

Optimizing the path dynamic response also includes the technology-specific dynamic response settings which are preset for different processing technologies (including tapping, roughing, smoothing) and can be activated in the part program by calling the respective dynamic response mode.

Free-form surface mode

Any fluctuation in curvature or torsion leads to a change in path velocity. This generally results in unnecessary decelerating and accelerating during the processing of free-form surface workpieces, which may adversely affect the quality of the surfaces of the workpieces.

The following functions are available for processing free-form surfaces.

- "Free-form surface mode: Basic functions"

  This makes the definition of the path velocity profile "less sensitive" to fluctuations in curvature and torsion.

- "Free-form surface mode: Extension function"

  This extension in standard LookAhead functionality is used to calculate the path velocity profile during the processing of free-form surfaces.

The advantages of free-form surface mode lie in a more homogeneous workpiece surface and lower machine load.

NC block compression

When a workpiece design is completed with a CAD/CAM system, the CAD/CAM system generally also compiles the corresponding part program to create the workpiece surface. To do so, most CAD/CAM systems use linear blocks to describe even curved sections of the workpiece surface. Many interpolation points are generally necessary to maintain the required contour accuracy. This results in many linear blocks, typically with very short paths.

The "NC block compressor" function uses polynomial blocks to perform a subsequent approximation of the contour specified by the linear blocks. During this process, an assignable number of linear blocks is replaced by a polynomial block. Furthermore, the number of linear blocks that can be replaced by a polynomial block also depends on the specified maximum permissible contour deviation and the contour profile.

Use of polynomial blocks provides the following advantages:

- Reduction of the number of required part program blocks for describing the workpiece contour
- Higher maximum path velocities

Combine short spline blocks

A spline defines a curve which is formed from 2nd or 3rd degree polynomials. With spline interpolation, the controller can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

References:
Programming Manual, Advanced, Special motion commands,
chapter: Spline interpolation

The advantages of the spline interpolation as compared to the linear interpolation are:

- Reduction of the number of required part program blocks for describing a curved contour.
- Soft, mechanical system-limiting curve characteristic also during transition between the part program blocks.

The disadvantages of the spline interpolation as compared to the linear interpolation are:

- For a spline curve no exact curve characteristic, but only a tolerance band can be specified, within which the spline curve should lie.

As with linear interpolation, the processing of splines can produce such short blocks that the path velocity must be reduced to enable interpolation of the spline blocks. This is alao the case, when the spline actually has a long, smooth curve. The "Combine short spline blocks" function allows you to combine these spline blocks such that the resulting block length is sufficient and does not reduce the path velocity.

Note

NC block compressor

The NC block compressor (COMPON, COMPCURV or COMPCAD) cannot be employed while compressing spline blocks since it can only be used to compress linear blocks.

3.2 Exact stop mode

Exact stop or exact stop mode

In exact stop traversing mode, all path axes and special axes involved in the traversing motion that are not traversed modally, are decelerated at the end of each block until they come to a standstill. The transition to the next block occurs only when all axes involved in the traversing motion have reached their programmed target position with subject to the selected exact stop criterion.

This results in the following response:

- The program run time is considerably longer compared to continuous path mode due to the deceleration of the axes and the wait time until "Exact stop" status is reached for all machine axes involved.
- In exact stop mode, undercuts can occur on the workpiece surface during machining.

Status "Exact stop"

The state of the machine axis based on the position difference relative to its position setpoint at the end of a traversing motion is also called an exact stop. The machine axis reaches the "exact stop" state, as soon as its following error is less than the specified position difference (exact stop criterion).

Application

Exact stop mode should always be used when the programmed contour must be executed exactly.

Activation

Exact stop mode can be activated on a modal basis or in blocks in the part program:

G command	Meaning
G60	Exact stop with **modal** effect
G9	Exact stop with **block-by-block** effect

Exact stop criteria "Exact stop coarse" and "Exact stop fine".

The exact stop criteria "Exact stop coarse" and "Exact stop fine" are used to specify tolerance windows for a machine axis reaching the "exact stop" state.

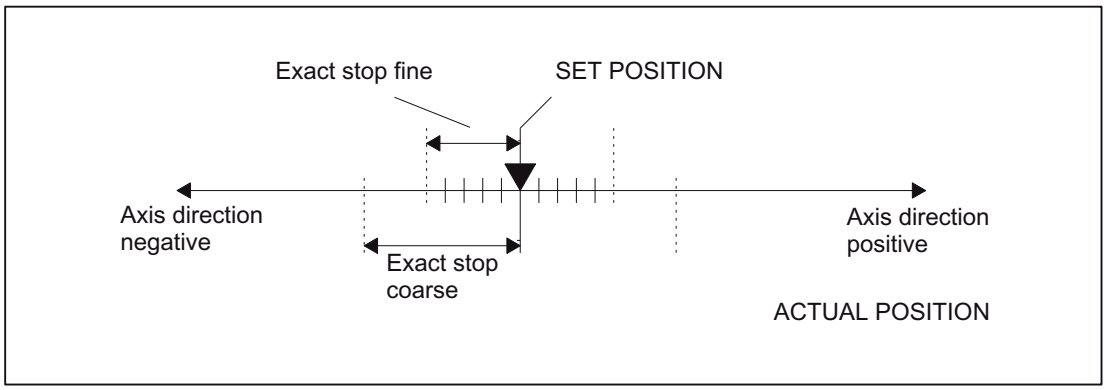


Figure 3-1 Tolerance windows of exact stop criteria

Parameters are assigned to the two exact stop criteria via the machine data:

MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)

MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)

Note

The tolerance windows of the exact stop criteria "Exact stop coarse" and "Exact stop fine" should be assigned in such a way that the following requirement is fulfilled:

"Exact stop coarse" > "Exact stop fine"

Exact stop criterion "Interpolator End"

In the case of the exact stop criterion "interpolator end" the block change to the next traversing block takes place, as soon as all path axes and special axes involved in the traversing motion, which do not traverse extending up to block, have reached their position according to set point programmed in the block. That is, the interpolator has executed the block.

The actual position and the following error of the relevant machine axes are not taken into consideration for exact stop criterion "Interpolator end". Thus, depending on the dynamic response of the machine axes, this can result in a relatively large smoothing of the contour at the block changes in comparison to the exact stop criteria "Exact stop coarse" and "Exact stop fine".

Activation of an exact stop criterion

An exact stop criterion is activated in the part program by programming the appropriate G command:

G command	Exact-stop criterion
G601	Exact stop fine
G602	Exact stop coarse
G603	Interpolator end

Block change depending on exact-stop criteria

The figure below illustrates the block change timing in terms of the selected exact stop criterion.

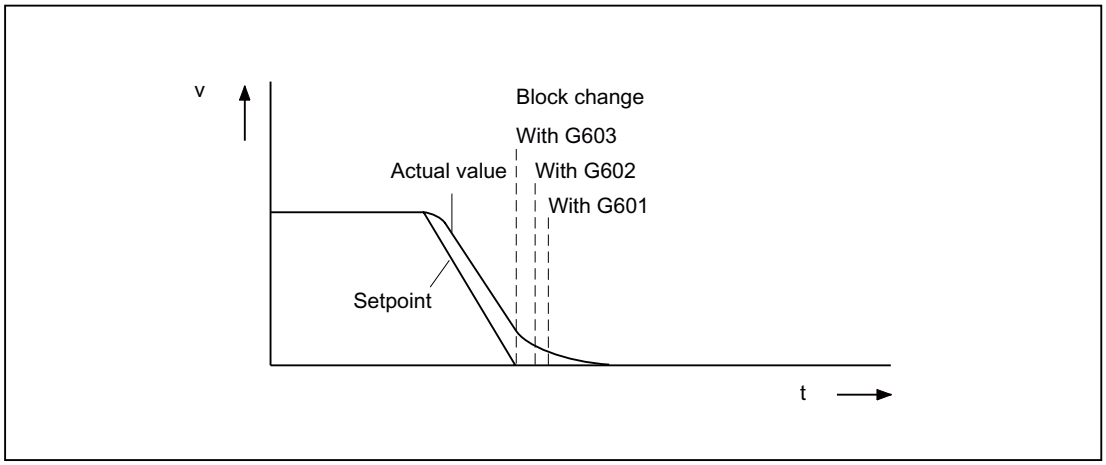


Figure 3-2 Block change accordance to selected exact stop criterion

Evaluation factor for exact stop criteria

A parameter set-dependent evaluation of the exact stop criteria can be specified via the following axis-specific machine data:

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

Applications:

- Adaptation of the positioning response to different mass ratios, such as after a gearshift
- Reduction in positioning time, depending on various machining states, such as roughing and finishing

Assignable specification of the active exact stop criterion

The active exact stop criterion can be permanently specified for the part program commands of the first G function group irrespective of the exact stop criterion programmed in the part program. This specification can be made independently for each of the following part program commands:

- Rapid traverse: G0
- Machining commands: G1, G2, G3, CIP, ASPLINE, BSPLINE, CSPLINE, POLY, G33, G34, G35, G331, G332, OEMIPO01, OEMIPO02, CT

The setting is done in a channel-specific manner via the following machine data:

MD20550 $MC_EXACT_POS_MODE (exact stop conditions for G0 and G1)

Coding

Each exact stop criterion is location-coded:

MD20550 $MC_EXACT_POS_MODE = <ZE>

- Ones position E: Rapid traverse
- Tens position Z: all other part program commands in the first G function group

Z or E	Active exact stop criterion
0	Programmed exact stop criterion
1	G601 (Exact stop window fine)
2	G602 (Exact stop window coarse)
3	G603 (Interpolator end)

Example

MD20550 $MC_EXACT_POS_MODE = 02

Ones position = 2:

With rapid traverse, exact stop criterion G602 (Exact stop window coarse) is always active, irrespective of any programming in the parts program.

Tens digit = 0:

For traversing with all other parts program commands of the first G function group, the exact stop criterion programmed in the parts program is active.

Assignable exact stop criterion for rapid traverse transitions in continuous path mode

The behavior at the block transition of part program blocks before and after rapid traverse blocks can be parameterized as follows:

MD20552 $MC_EXACT_POS_MODE_G0_TO_G1 = <value>

Value	Meaning
0	No additional stop at the block transition.
1	Stop at block transition: Same behavior as in the case of G601 (Exact stop window fine)
2	Stop at block transition: Same behavior as G602 (Exact stop window coarse).
3	Stop at block transition: Same behavior as G603 (Interpolator end).
4	Like 0, in addition, the override of the next non-G0 block is taken into account with LookAhead in the G0 block during the transition from G0 to non-G0.
5	Like 0; in addition, the override of the next block is taken into account with LookAhead during the transition from G0 to non-G0 and from non-G0 to G0.

3.3 Continuous-path mode

3.3.1 General functionality

Continuous-path mode

In the continuous-path mode, the path velocity is **not** decelerated for the block change in order to permit the fulfillment of an exact stop criterion. The objective of this mode is to avoid rapid deceleration of the path axes at the block-change point so that the axis velocity remains as constant as possible when the program moves to the next block. To achieve this objective, the "LookAhead" function is also activated when the continuous-path mode is selected.

Continuous-path mode causes the smoothing and tangential shaping of angular block transitions by local changes in the programmed contour. The extent of the change relative to the programmed contour can be limited by specifying the overload factor or rounding criteria.

Continuous-path mode:

- Contour rounding.
- Reduces machining times by eliminating braking and acceleration processes that are required to fulfill the exact-stop criterion.
- Improves cutting conditions because of the more constant velocity.

Continuous-path mode is suitable if:

- A contour must be traversed as quickly as possible (e.g. with rapid traverse).
- The exact contour may deviate from the programmed contour within a specific tolerance for the purpose of obtaining a continuous contour

Continuous-path mode is not suitable if:

- A contour is to be traversed precisely.
- An absolutely constant velocity is required.

Implicit exact stop

In some cases, an exact stop needs to be generated in continuous-path mode to allow the execution of subsequent actions. In such situations, the path velocity is reduced to zero.

- If auxiliary functions are output before the traverse motion, the previous block is only terminated when the selected exact-stop criterion is fulfilled.
- If auxiliary functions are to be output after the traverse motion, they are output after the interpolator end of the block.
- If an executable block (e.g. starting of a positioning axis) contains no travel information for the path axes, the previous block is terminated on reaching the selected exact-stop criterion.
- If a positioning axis is declared to be the geometry axis, the previous block is terminated at the interpolator end when the geometry axis is programmed.
- If a synchronized axis is programmed that was last programmed as a positioning axis or spindle (initial setting of the special axis is positioning axis), the previous block is ended at the interpolator end.
- If the transformation is changed, the block previously processed is terminated with the active exact-stop criterion.
- A block is terminated on interpolator end if the following block contains the changeover of the acceleration profile BRISK/SOFT (see Section "B2: Acceleration (Page 245)").
- If the "empty buffer" function is programmed, the previous block is terminated when the selected exact-stop criterion is reached.

Velocity = 0 in continuous-path mode

Regardless of the implicit exact stop response, the path motion is braked down to zero velocity at the end of the block in cases where:

- Positioning axes are programmed with the instruction `POS`, and their traversing time exceeds that of the path axes. The block change occurs when the "exact stop fine" of the positioning axes is reached.
- The time taken to position a spindle programmed with the instruction `SPOS` is longer than the traversing time of the path axes. The block change is carried out when the "exact stop fine" of the positioning spindle is reached.
- The current block contains traversing commands for geometry axes and the following block traversing commands for synchronized axes or, alternatively, the current block contains traversing commands for synchronized axes and the subsequent block traversing commands for geometry axes.
- Synchronization is required.

3.3.2 Velocity reduction according to overload factor

Function

The function lowers the path velocity in continuou-path mode until the non-tangential block transition can be traversed in one interpolation cycle while respecting the deceleration limit and taking an overload factor into account.

With the reduced velocity, axial jumps in velocity are produced with a non-tangential contour at the block transition. These jumps in velocity are also performed by the coupled motion synchronized axes. The jump in velocity prevents the path velocity dropping to zero. This jump is performed if the axial velocity was reduced with the axial acceleration to a velocity from which the new setpoint can be reached with the jump. The magnitude of the setpoint jump can be limited using an overload factor. Because the magnitude of the jump is axial, the minimum jump of the path axes which are active during the block change is considered during block transition.

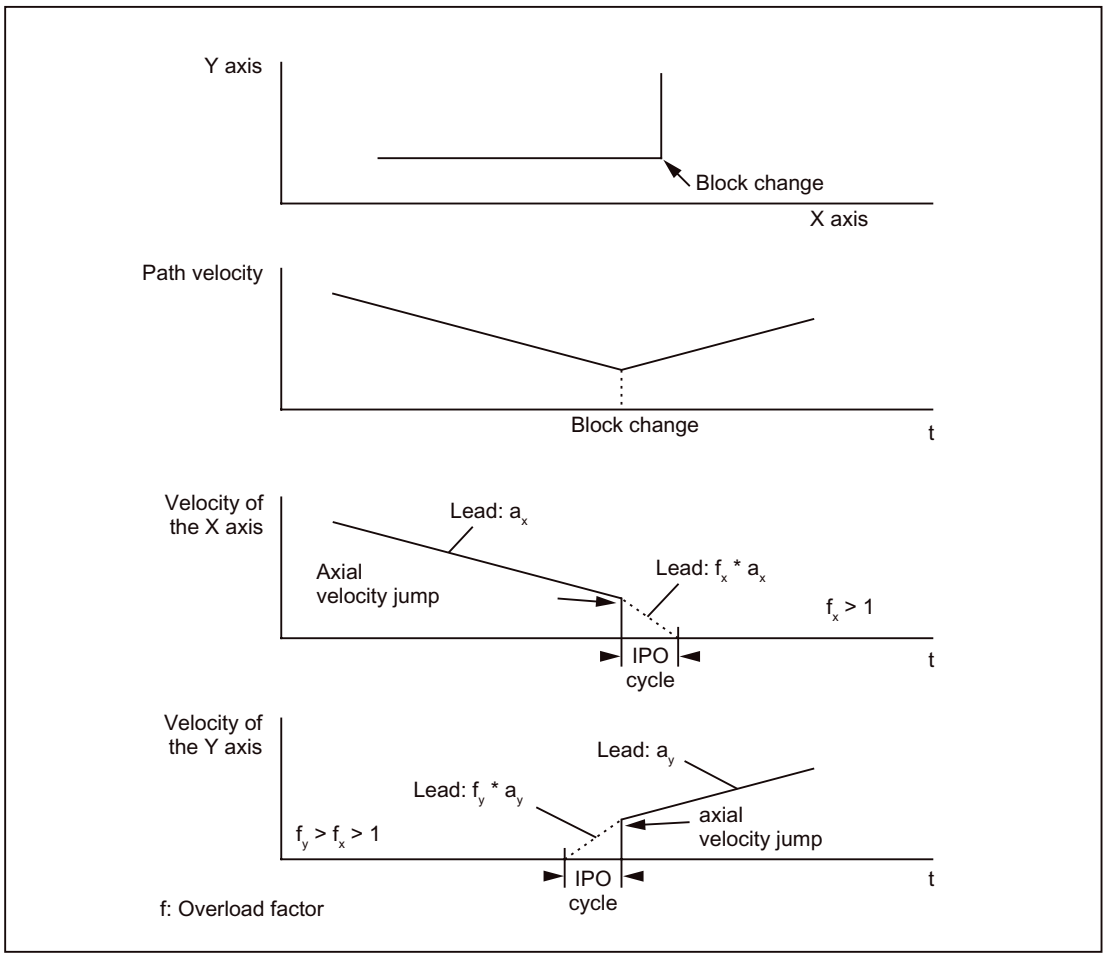


Figure 3-3 Axial velocity change on block transition

With a practically tangential block transition, the path velocity is not reduced if the permissible axial accelerations are not exceeded. This means that very small bends in the contour (e.g. 0.5°) are overtraveled directly.

Overload factor

The overload factor restricts step changes in the machine axis velocity at block ends. To ensure that the velocity jump does not exceed the maximum load on the axis, the jump is derived from the acceleration of the axis.

The overload factor indicates the extent by which the acceleration of the machine axis (MD32300 $MA_MAX_AX_ACCEL) may be exceeded for an IPO-cycle.

The velocity jump results as follows:

Velocity jump = axis acceleration * (overload factor-1) * interpolator cycle.

The overload factor is saved in the machine data:

MD32310 $MA_MAX_ACCEL_OVL_FACTOR (overload factor for axial velocity jumps)

Factor 1.0 means that only tangential transitions with finite velocity can be traversed. For all other transitions, the velocity is reduced to zero by changing the setpoint. This behavior is equivalent to the function "Exact stop with interpolator end". This is undesirable for continuous-path mode, so the factor must be set to greater than 1.0.

Note

For startup and installation, please note that the factor must be reduced if the machine is likely to be subject to vibrations during angular block transitions and rounding is not to be used.

By setting the following machine data, the block transitions are rounded independent of the set overload factor with G641/G642:

MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS

Activation/deactivation

Continuous-path mode with a reduction in speed according to the overload factor can be activated in any NC part program block by the modal command `G64`.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode G64 can be deactivated by selecting:

- Modal exact stop G60
- Rounding G641, G642, G643, G644 or G645

Implicit continuous-path mode

If it is not possible to insert rounding blocks in continuous-path mode with rounding G641 due to the very short block path lengths (e.g. zero-clocked blocks), the mode is switched over to continuous-path mode G64.

3.3.3 Rounding

Function

Rounding means that an angular block transition is changed to a tangential block transition by a local change to the programmed contour. This gives the area in the vicinity of the original angular block transition (including transitions between intermediate blocks inserted by the CNC) a continuous contour.

During rounding, it is not only the geometry axes that are taken into account, but all machine axes which traverse synchronously. The rounding function therefore smoothes the traversing path of orientation axes as well as general velocity step changes in synchronized axes.

Note

Rounding cannot and should not replace the functions for defined smoothing, i.e. RND, RNDM, ASPLINE, BSPLINE, CSPLINE.

If a rounding motion initiated by G641, G642, G643, G644 or G645 is interrupted, the corner point of the original contour will be used for subsequent repositioning, rather than the interruption point.

Synchronization

Rounding involves shortening discontinuously adjoining blocks and inserting one or two intermediate blocks at this point. The original block boundary is removed and can no longer be used for synchronization conditions (e.g. auxiliary function output parallel to motion, stop at end of block).

With rounding, all synchronization conditions are best referred to the end of the shortened first block and not to the end of the intermediate rounding block. The following block is thus not started and with a stop at end of block, the contour of the following block can still be changed.

Execution

Rounding is only performed if the block transition is to be traveled with finite velocity. The maximum path speed is influenced by the curvature. The maximum acceleration values of the axes are not exceeded. A block without traverse information for the path axes requires velocity "zero" and therefore no rounding.

Rounding is also used if the traversal of the block transition requires a velocity that lies below the permissible velocity at the end of the block according to G64 (see Section "Velocity reduction according to overload factor (Page 178)", paragraph "Overload factor").

No intermediate rounding blocks

An intermediate rounding block is not inserted in the following cases:

- The axis stops between the two blocks.

  This occurs when:

  – The following block contains an auxiliary function output before the motion.

  – The following block does not contain a path motion.

  – An axis is traversed for the first time as a path axis for the following block when it was previously a positioning axis.

  – An axis is traversed for the first time as a positioning axis for the following block when it was previously a path axis.

  – The previous block traverses geometry axes and the following block does not.

  – The following block traverses geometry axes and the previous block does not.

  – Before tapping, the following block uses `G33` as preparatory function and the previous block does not.

  – A change is made between BRISK and SOFT.

  – Axes involved in the transformation are not completely assigned to the path motion (e.g. for oscillation, positioning axes).

- The rounding block would slow down the part program execution.

  This occurs:

  – Between two very short blocks.

    Since each block requires at least one interpolation cycle, the added intermediate block would double the machining time.

  – If a block transition with G64 (continuous-path mode without rounding) can be traversed without a reduction in velocity.

    Corner rounding would increase the machining time. This means that the value of the permitted overload factor (MD32310 $MA_MAX_ACCEL_OVL_FACTOR) affects whether a block transition is rounded or not. The overload factor is only taken into account for rounding with G641 / G642. The overload factor has no effect in the case of rounding with G643 (this behavior can also be set for G641 and G642 by setting MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS = TRUE).

- Rounding is not parameterized.

  This occurs when:

  – For G641 ADISPOS= 0 in G0 blocks (default!).

  – For G641 ADIS = 0 in non-G0 blocks (default!).

  – For transition from G0 to non-G0 or non-G0 to G0, the smaller of the values for ADISPOS and ADIS will apply in the case of G641.

  – For G642/G643, all axis-specific tolerances are zero.

- The block does not contain traversing motion (zero block).

  This occurs when:

  – Synchronized actions are active.

    Normally, the Interpreter eliminates zero blocks. However, if synchronized actions are active, this zero block is included and also executed. In so doing, an exact stop is initiated corresponding to the active programming. This allows the synchronized action to also switch.

  – Zero blocks are generated by program jumps.

Synchronized axes

If a number of paths need to be synchronized (e.g. contour, special axis), then every path must always have its own rounding area.

There are no practical means of achieving this exactly. Therefore, on the basis of the specific meaning of the contour (geometry axis), the following procedure is applied:

Rounding behavior with synchronized paths

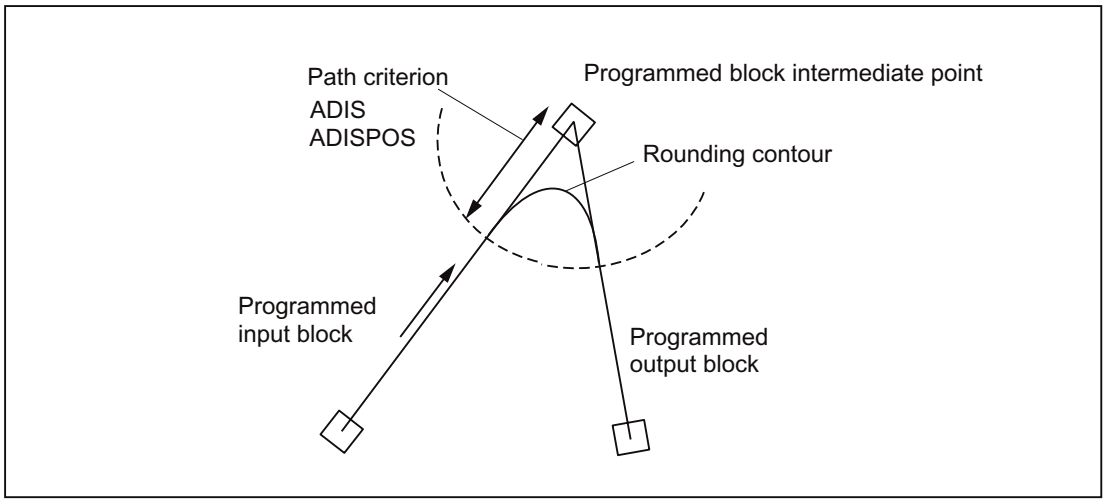
Original path for:		Rounding path result:
Geometry axes	Orientation/ synchronized axis	
Smooth	Smooth	Defined path is traversed exactly.
Smooth	Angular	Intermediate blocks, the geometry axes follow the path exactly, all orientation/synchronized axis paths are smoothed.
Angular	Smooth	Intermediate block, the geometry axes perform rounding, all orientation/synchronized axis paths are smoothed.
Angular	Angular	Intermediate block, the geometry axes perform rounding, all orientation/synchronized axis paths are smoothed.

3.3.3.1 Rounding according to a path criterion (G641)

Function

In continuous-path mode with rounding according to a path criterion, the size of the rounding area is influenced by the path criteria ADIS and ADISPOS.

The path criteria ADIS and ADISPOS describe the maximum distances which a rounding block can occupy before and after a block.



Note

Acute angles produce rounding curves with a large degree of curvature and therefore cause a corresponding reduction in velocity.

Note

ADISPOS is programmed in the same way as ADIS, but must be used specifically for movements in rapid traverse mode G00.

Scope of the path criterion

- ADIS or ADISPOS must be programmed. If the default is "zero", G641 behaves like G64.
- If only one of the blocks involved is rapid traverse G0, the smaller rounding distance applies.
- If a very small value is used for ADIS, the controller must make sure that every interpolated block, even an intermediate rounding block, contains at least one interpolation point. The maximum path velocity is thereby limited to ADIS / interpolation cycle.
- Irrespective of ADIS and ADISPOS, the rounding area is limited by the block length.

  In blocks with short distances (distance < 4* ADIS and < 4 * ADISPOS respectively), the rounding distance is reduced so that a traversable part of the original block is retained. The remaining length depends on the axis path and is approximately 60% of the distance still to be traversed in the block. ADIS or ADISPOS is therefore reduced to the remaining 40% of the distance to be traversed. This algorithm prevents a rounding block being inserted for a very small change in contour. In this case, switchover to continuous-path mode G64 is automatic until rounding blocks can be inserted again.

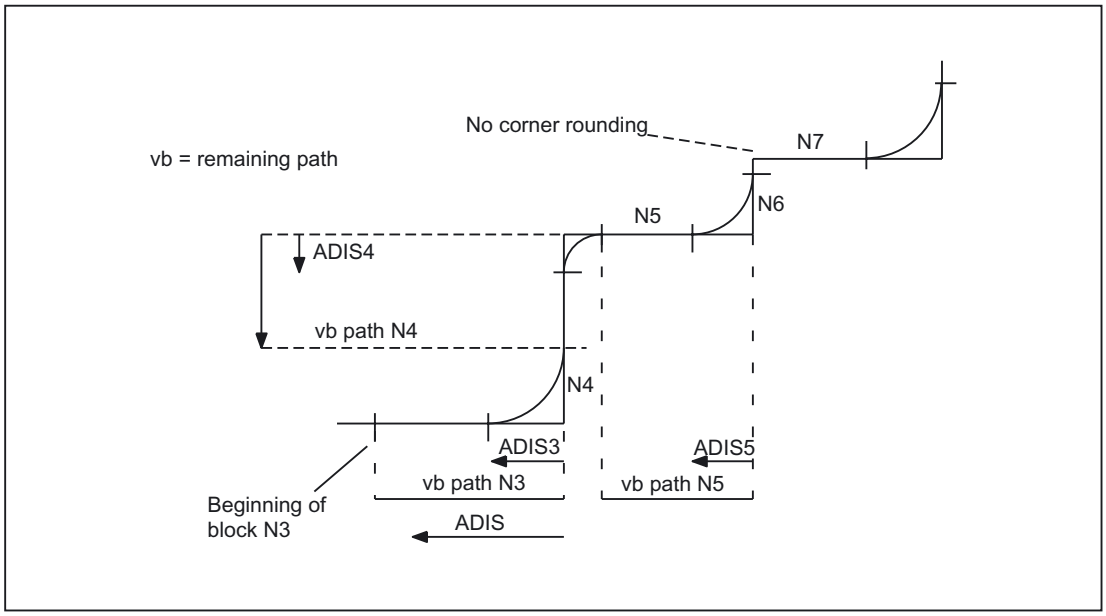


Figure 3-4 Path with limitation of ADIS

Activation/deactivation

Continuous-path mode with rounding based on a path criterion can be activated in any NC part program block by the modal command G641. Before or on selection, the path criteria ADIS/ADISPOS must be specified.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding based on a path criterion (G641) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G642, G643, G644 or G645

Program example

```
Program code                          Comment
N1 G641 Y50 F10 ADIS=0.5              ; Continuous-path mode with rounding based on a
                                        path criterion (rounding clearance: 0.5 mm)
N2 X50
N3 X50.7
N4 Y50.7
N5 Y51.4
N6 Y51.0
N7 X52.1
```

3.3.3.2 Rounding in compliance with defined tolerances (G642/G643)

Function

In continuous-path mode involving rounding in compliance with defined tolerances, the rounding normally takes place while adhering to the maximum permissible path deviation. Instead of these axis-specific tolerances, the maintenance of the maximum contour deviation (contour tolerance) or the maximum angular deviation of the tool orientation (orientation tolerance) can be configured.

Activation

Continuous-path mode with rounding in compliance with defined tolerances can be activated in any NC part program block by the modal command G642 or G643.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding in compliance with defined tolerances (G642/G643) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G641, G644 or G645

Differences between G642 - G643

With regard to their rounding behavior, commands G642 and G643 differ as follows:

G642	G643
With G642, the rounding path is determined on the basis of the shortest distance for rounding all axes. This value is taken into account when generating a rounding block.	In the case of G643, each axis may have a different rounding path. The rounding travels are taken into account axis-specifically and block-internally (⇒ no separate rounding block).
With G642, the rounding area results from the smallest tolerance setting.	Very different specifications for the contour tolerance and the tolerance of the tool orientation can only have effect with G643.

Parameterization

Maximum path deviation

The maximum path deviation permitted with G642/G643 is set for each axis in the machine data:

MD33100 $MA_COMPRESS_POS_TOL

Contour tolerance and orientation tolerance

The contour tolerance and orientation tolerance are set in the channel-specific setting data:

SD42465 $SC_SMOOTH_CONTUR_TOL (maximum contour deviation)

SD42466 $SC_SMOOTH_ORI_TOL (maximum angular deviation of the tool orientation)

The settings data can be programmed in the NC program and can in this way be specified differently for each block transition.

Note

The setting data SD42466 $SC_SMOOTH_ORI_TOL is effective only in active orientation transformation.

Rounding behavior

Rounding behavior with G642 and G643 is configured via the machine data:

MD20480 $MC_SMOOTHING_MODE (rounding behavior with G64x)

The units positions (**E**) define the behavior for G643, the tens positions (**Z**) the behavior for G642:

Value E or Z	Meaning
0	**All axes:** Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL
1	**Geometry axes:** Rounding by maintaining the contour tolerance: SD42465 $SC_SMOOTH_CONTUR_TOL **Remaining axes:** Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL
2	**Geometry axes:** Rounding by maintaining the orientation tolerance: SD42466 $SC_SMOOTH_ORI_TOL **Remaining axes:** Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL
3	**Geometry axes:** Rounding by maintaining the contour tolerance and the orientation tolerance: SD42465 $SC_SMOOTH_CONTUR_TOL SD42466 $SC_SMOOTH_ORI_TOL **Remaining axes:** Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL
4	**All axes:** The rounding length programmed with `ADIS` or with `ADISPOS` is used (as in case of G641). Any axis-specific tolerance or contour and orientation tolerance specifications are ignored.

Profile for limit velocity

The use of a velocity profile for rounding in compliance with defined tolerances is controlled via the hundreds position in MD20480:

Value	Meaning	
< 100:	A profile of the limit velocity is calculated within the rounding area, based on the defined maximum values for acceleration and jerk on the participating axes or path. This can lead to an increase in the path velocity in the rounding area and therefore to the acceleration of the participating axes.	
≥100:	A profile of the limit velocity is not calculated for rounding blocks with G641/G642. A constant velocity limit is specified instead. This prevents the participating axes being accelerated into the rounding area during rounding with G641/G642. However, in certain cases, this setting can cause the rounding blocks to be traversed too slowly, especially in large rounding areas.	
	1xx:	No velocity profile for G641
	2xx:	No velocity profile for G642

Note

MD28530 $MC_MM_PATH_VELO_SEGMENTS (number of memory elements for limiting the path velocity)

Supplementary conditions

Restriction for protection zones with active radius compensation and tool orientation:

Although tool radius compensation is applied for a tool orientation, which is not perpendicular to one of the three datum planes of the basic coordinate system, the protection zones are not rotated onto the corresponding plane.

For G643 the following must apply:

MD28530 $MC_MM_PATH_VELO_SEGMENTS > 0 (number of memory elements for limiting the path velocity)

If this condition is met, then it must be applicable for all axes:

MD35240 $MC_ACCEL_TYPE_DRIVE = FALSE (acceleration characteristic DRIVE for axes on/off)

3.3.3.3 Rounding with maximum possible axial dynamic response (G644)

Function

Maximizing the dynamic response of the axes is key to this type of continuous-path mode with rounding.

Note

Rounding with G644 is only possible if:

- All the axes involved contain only a linear motion in both the observed blocks.
- **No** kinematic transformation is active

In case an involved axis contains a polynomial (polynomial programmed, spline active, compressor active) or a kinematic transformation is active, the block transition is rounded with G642.

Activation

Continuous-path mode with rounding with the maximum possible axial dynamic response can be activated in any NC part program block by the modal command `G644`.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding with the maximum possible axial dynamic response (G644) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G641, G642, G643 or G645

Parameterization

Rounding behavior with G644 is configured via the thousands and tens of thousands places in the machine data:

MD20480 $MC_SMOOTHING_MODE (rounding behavior with G64x)

Value	Meaning
Thousand's place:	
0xxx:	When rounding with G644, the maximum deviations for each axis specified by the following machine data are respected: MD33100 $MA_COMPRESS_POS_TOL If the dynamics of the axis permit, then any specified tolerance is not utilized.
1xxx:	Input the maximum rounding path by programming ADIS=... or ADISPOS=...(as for G641)
2xxx:	Input the maximum possible frequencies of each axis in the rounding area using the machine data: MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead) The rounding area is defined so that no frequencies in excess of the specified maximum can occur while the rounding motion is in progress.
3xxx:	Any axis that has a velocity jump at a corner traverses around the corner with the maximum possible dynamic response (maximum acceleration and maximum jerk). SOFT: If SOFT is active, the maximum acceleration **and** the maximum jerk of each axis is maintained. BRISK: If BRISK is active, only the maximum acceleration and **not** the maximum jerk of each axis is maintained. With this setting, neither the maximum deviations nor the rounding distance are checked. The resulting deviations or rounding distances are determined exclusively by the dynamic limits of the respective axis and the current path velocity.
4xxx:	As in case of 0xxx, the maximum deviations of each axis specified with the following machine data are used: MD33100 $MA_COMPRESS_POS_TOL Contrary to 0xxx, the specified tolerance is also utilized, if possible. Therefore, the axis does not attain its maximum possible dynamics.
5xxx:	As in case of 1xxx, the maximum possible rounding path is specified through programming of ADIS=... or ADISPOS= respectively. Contrary to 1xxx, the specified rounding path is also utilized, if possible. Therefore, the axes involved do not attain their maximum possible dynamics.
Ten thousands digit	
0xxxx	The velocity profiles of the axes in the rounding area are determined without jerk limiting for BRISK and with jerk limiting for SOFT.
1xxxx	The velocity profiles of the axes in the rounding area are always determined with jerk limiting, regardless of whether BRISK or SOFT is active.

When specifying the maximum axial deviations (MD33100 $MA_COMPRESS_POS_TOL) or the maximum rounding distance (ADIS / ADISPOS) the available rounding path is normally not used, if permitted by the dynamics of the axes involved. Through this, the length of the rounding path depends on the active path feedrate. In case of lower path speeds, one gets lower deviations from the programmed contours. However, it can be set that in these cases the specified maximum axial deviation or the specified rounding distance is utilized, if possible. In this case the deviations depend on the programmed contour independent of the programmed path feedrate.

Note

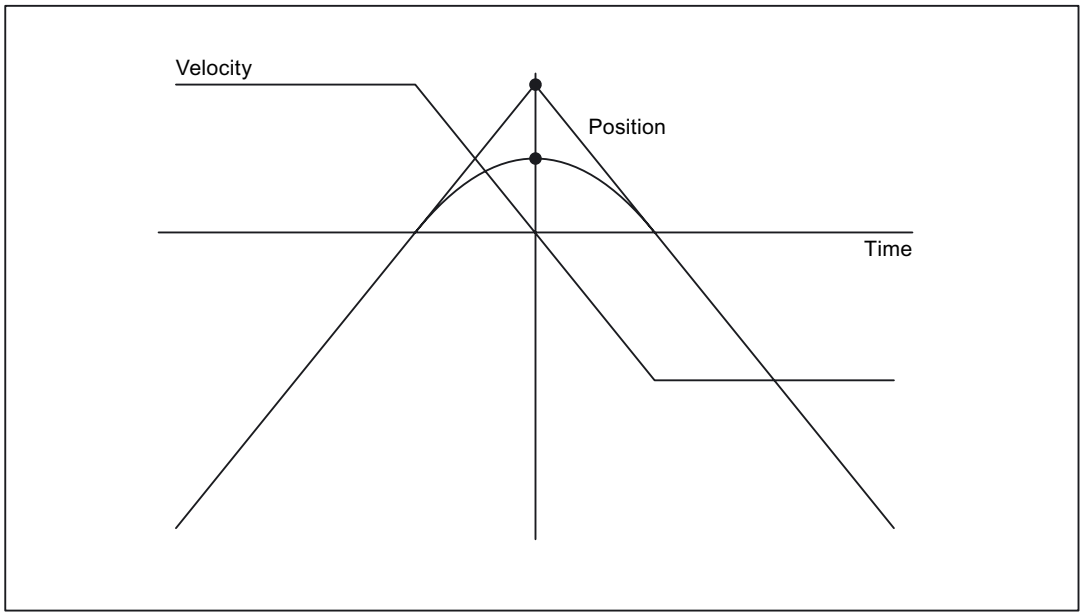
Apart from the ones mentioned, the following limitation can also become active additionally:

The rounding distance cannot exceed half the length of the original participating blocks.

Jerk limitation

The smoothing of the velocity jump on each axis and thus the shape of the rounding path depends on whether an interpolation is performed with or without jerk limitation.

Without jerk limitation the acceleration of each axis reaches its maximum value in the entire rounding area.



With jerk limitation, the jerk of each axis is limited to its maximum value within the rounding area. The rounding motion thus generally consists of three phases:

- **Phase 1**

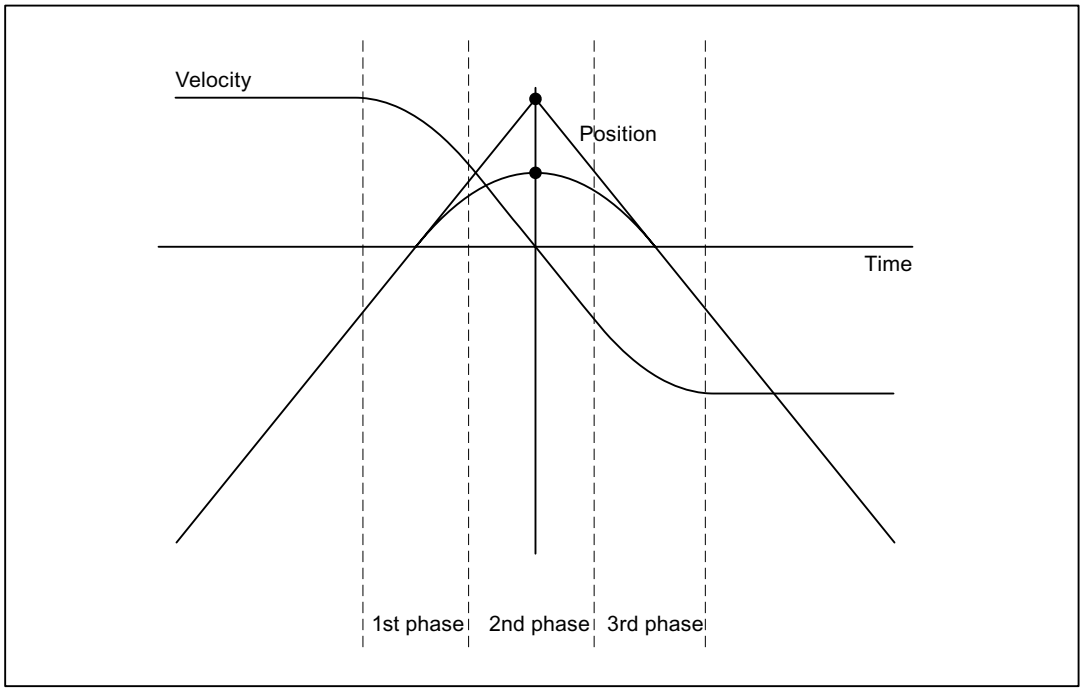
  During phase 1, each axis builds up its maximum acceleration. The jerk is constant and equal to the maximum possible jerk on the respective axis.

- **Phase 2**

  During phase 2, the maximum permissible acceleration is applied.

- **Phase 3**

  During phase 3, which is the last phase, the acceleration of each axis is reduced back to zero with the maximum permissible jerk.



3.3.3.4 Rounding of tangential block transitions (G645)

Function

In continuous-path mode with rounding, rounding blocks are also only generated on tangential block transitions if the curvature of the original contour exhibits a jump in at least one axis.

The rounding motion is defined here so that the acceleration of all axes involved remains smooth (no jumps) and the parameterized maximum deviations from the original contour (MD33120 $MA_PATH_TRANS_POS_TOL) are not exceeded.

In the case of angular, non-tangential block transitions, the rounding behavior is the same as with G642 (see Section "Rounding in compliance with defined tolerances (G642/G643) (Page 185)").

Activation/deactivation

Continuous-path mode with rounding of tangential block transitions can be activated in any NC part program block by the modal command `G645`.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding of tangential block transitions (G645) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G641, G642, G643 or G644

Comparison between G642 and G645

When rounding with G642, the only block transitions rounded are those which form a corner, i.e. the velocity of at least one axis jumps. However, if a block transition is tangential, but there is a jump in the curvature, no rounding block is inserted with G642. If this block transition is traversed with finite velocity, the axes experience some degree of jump in acceleration which (with the jerk limit activated!) may not exceed the parameterized limit (MD32432 $MA_PATH_TRANS_JERK_LIM). Depending on the level of the limit, the path velocity at the block transition may be greatly reduced as a result. This constraint is avoided by using G645 because the rounding motion is defined here in such a way that no jumps occur in acceleration.

Parameterization

The following machine data indicates the maximum permissible path deviation for each axis during rounding with G645:

MD33120 $MA_PATH_TRANS_POS_TOL

This value is only of relevance to tangential block transitions with variable acceleration. When angular, non-tangential block transitions are rounded, (as with G642) the tolerance from MD33100 $MA_COMPRESS_POS_TOL becomes effective.

See also

Free-form surface mode: Basic functions (Page 222)

3.3.4 LookAhead

3.3.4.1 Standard functionality

Function

LookAhead is a function which is active in continuous-path mode (G64, G64x) and determines a foreseeable velocity control for multiple NC part program blocks over and beyond the current block.

Note

LookAhead is only available for path axes, **not** for spindles and positioning axes.

If a part program contains consecutive blocks with very small paths, only one velocity is reached per block without LookAhead, enabling deceleration of the axes at the end of the block while maintaining acceleration limits. This means that the programmed velocity is not reached at all. With LookAhead, however, it is possible to implement the acceleration and deceleration phase over multiple blocks with approximately tangential block transitions, thereby achieving a higher feedrate with shorter distances.

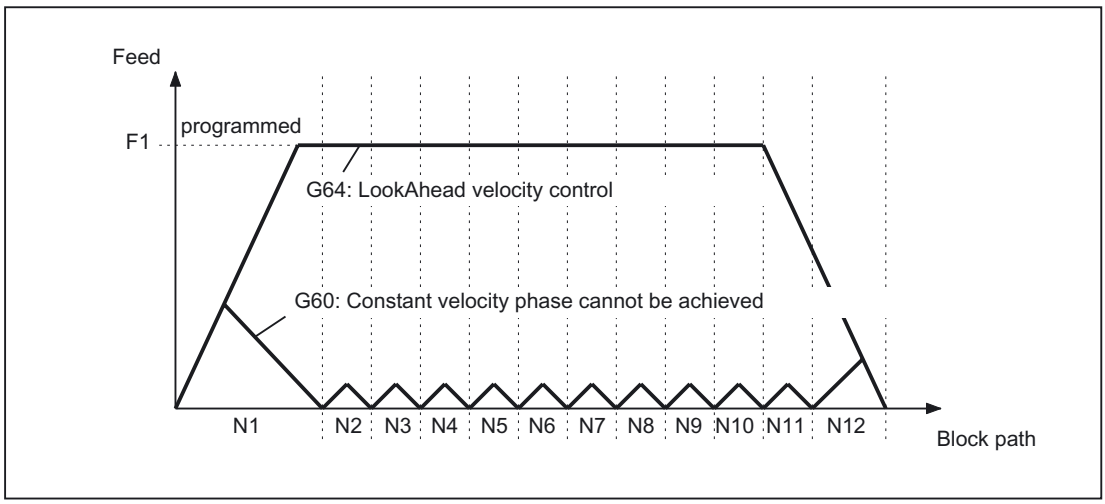


Figure 3-5 Velocity control with short distances and exact stop G60 or continuous-path mode G64 with LookAhead

Deceleration to velocity limits is possible with LookAhead such that violation of the acceleration and velocity limit is prevented.

LookAhead takes plannable velocity limits into consideration such as:

- Exact stop at block end
- Velocity limit in the block
- Acceleration limit in the block
- Velocity limit on block transition
- Synchronization with block change at block transition.

Mode of operation

LookAhead carries out a block-specific analysis of velocity limits and specifies the required brake ramp profile based on this information. LookAhead is adapted automatically to block length, braking capacity and permissible path velocity.

For safety reasons, the velocity at the end of the last prepared block must initially be assumed to be zero because the next block might be very small or be an exact-stop block, and the axes must have been stopped by the end of the block.

With a series of blocks with high set velocity and very short paths, the speed can be increased in each block depending on the velocity value currently calculated by the LookAhead function in order to achieve the required set velocity. After this it can be reduced so that the velocity at the end of the last block considered by the LookAhead function can be zero. This results in a serrated velocity profile (see the following fig.) which can be avoided by reducing the set velocity or increasing the number of blocks considered by the LookAhead function.

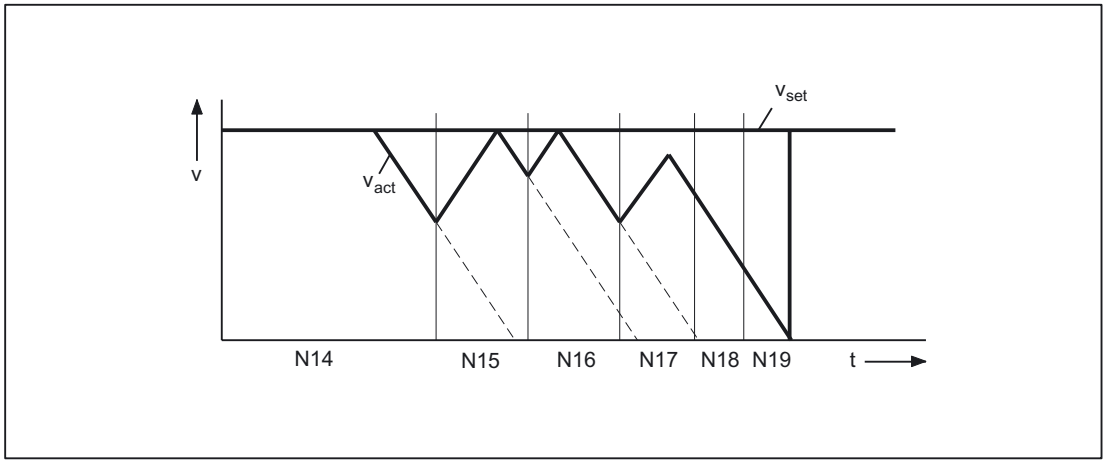


Figure 3-6 Example for modal velocity control (number of blocks considered by the LookAhead function = 2)

Activation/deactivation

LookAhead is activated by selecting continuous-path mode G64, G641, G642, G643, G644 or G645.

Selecting the exact stop which works on a non-modal basis enables rounding to be interrupted (G09).

LookAhead is deactivated by selecting the modal exact stop (G60).

Parameterization

Number of blocks

To achieve reliable axis traversal in continuous-path mode, the feedrate must be adapted over several blocks. The number of blocks considered by the LookAhead function is calculated automatically and can, if required, be limited by a machine data. The default setting is "1", which means that LookAhead only considers the following block for velocity control.

Because LookAhead is especially important for short blocks (relative to the deceleration path), the number of blocks required is of interest for LookAhead braking. It is sufficient to consider the path length to be equal to the deceleration path that is required to brake from maximum velocity to standstill.

For a machine with a low axial acceleration of a = 1 m/s^2 and a high feedrate of v_{path} = 10 m/min, the following number of $n_{LookAhead}$ blocks are allocated to the controller where it has has an attainable block cycle time of TB = 10 ms:

$n_{LookAhead}$ = Deceleration path/Block length = (v_{path}^2 / (2a)) / (v_{path} * TB) = 9

Considering these conditions, it is advisable to adapt the feedrate over 10 blocks. The number of blocks entered for the LookAhead function forecast does not change the LookAhead algorithm and memory requirement.

Since the machining velocity is very often set to a lower value than the maximum velocity in a program, more blocks than are required would be predicted, overloading the processor unnecessarily. For this reason, the required number of blocks is derived from the velocity which is calculated from the following multiplication:

- Programmed velocity * MD12100 $MN_OVR_FACTOR_LIMIT_BIN

  (when using a **binary**-coded feedrate override switch)

- Programmed velocity * MD12030 $MN_OVR_FACTOR_FEEDRATE[**30**]

  (when using a **gray**-coded feedrate override switch)

The value for MD12100 or the 31st override value for MD12030 defines the dynamic response reserves which the velocity control provides for when the path feed is overshot.

Note

The 31st override value for MD12030 should correspond to the highest override factor which is actually used.

Note

The number of blocks considered by the LookAhead function is limited by the possible number of NC blocks in the IPO buffer.

Velocity profiles

In addition to the fixed, plannable velocity limitations, LookAhead can also take account of the programmed velocity. This makes it possible to achieve a lower velocity by applying LookAhead beyond the current block.

- **Determination of the following block velocity**

  One possible velocity profile contains the determination of the following block velocity.

  Using information from the current and the following NC block, a velocity profile is calculated from which, in turn, the required velocity reduction for the current override is derived.

  The calculated maximum value of the velocity profile is limited by the maximum path velocity.

  With this function it is possible to initiate a speed reduction in the current block taking override into account such that the lower velocity of the following block can be achieved. If the reduction in velocity takes longer than the travel time of the current block, the velocity is further reduced in the following block. Velocity control is only ever considered for the following block.

  The function is activated via the machine data:

  MD20400 $MC_LOOKAH_USE_VELO_NEXT_BLOCK

Value	Meaning
TRUE	Function active
FALSE	Function **not** active

- **Definition of override points**

  If the velocity profile of the following block velocity is not sufficient because, for example, very high override values (e.g. 200%) are used or a constant cutting rate G96/G961 is active, with the result that the velocity must be further reduced in the following block, LookAhead provides a way of reducing the programmed velocity over several NC blocks.

  By defining override points, LookAhead calculates a limiting velocity profile for each value. The required velocity reductions for the current override are derived from these profiles.

  The calculated maximum value of the velocity profile is limited by the maximum path velocity.

  The upper point should cover the velocity range that will be reached by the maximum value set in the machine data:

  MD12030 $MN_OVR_FACTOR_FEEDRATE [n] (evaluation of the path feedrate override switch)

  It can also be reached via the value of the machine data:

  MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

  In this way, a reduction of the velocity continuing into the block in which it is programmed can be avoided.

  If velocity reductions across block boundaries are required at a 100% override, a point should be set in the lower override range as well.

  The number of override points used per channel is specified in the machine data:

  MD20430 $MC_LOOKAH_NUM_OVR_POINTS (number of override switch points for LookAhead)

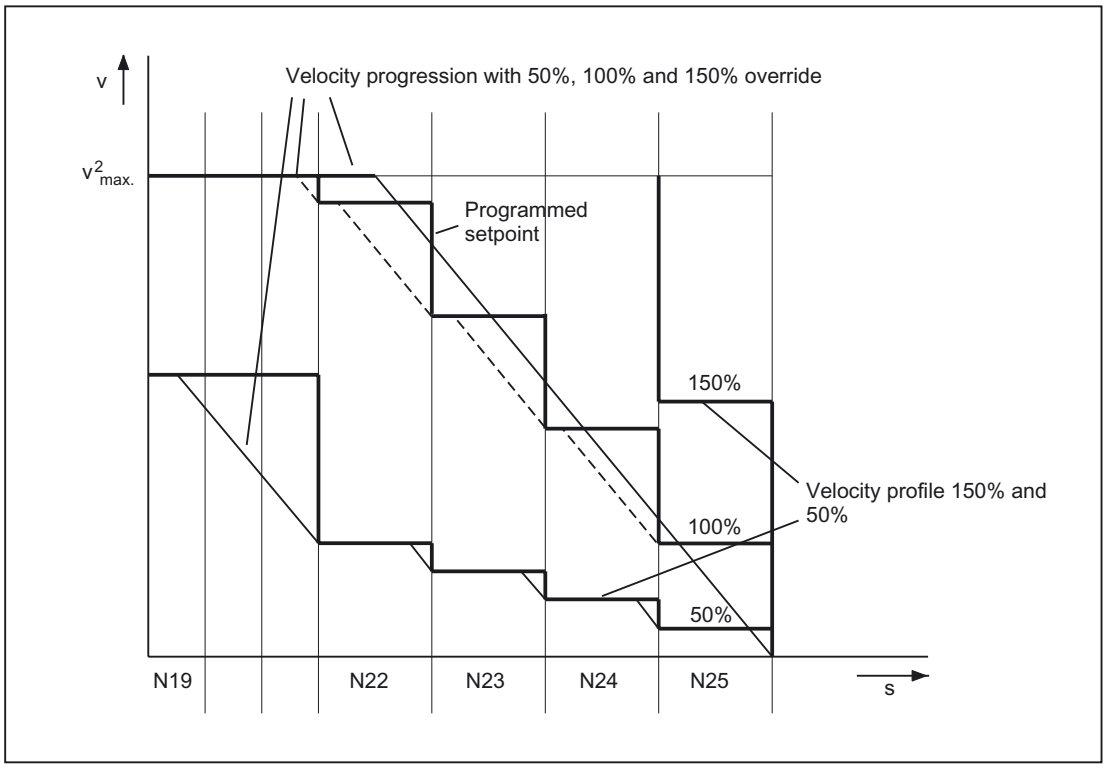
  The associated points are stored in the machine data:

  MD20440 $MC_LOOKAH_OVR_POINTS (prepared override velocity characteristics with LookAhead)

  Example:

  Limiting velocity characteristics, whereby:

  – Override = 50%, 100% or 150%
  – Number of LookAhead blocks = 4
  – MD20430 $MC_LOOKAH_NUM_OVR_POINTS = 2
  – MD20440 $MC_LOOKAH_OVR_POINTS = 1.5, 0.5
  – MD20400 $MC_LOOKAH_USE_VELO_NEXT_BLOCK = 1



A combination of both procedures (determination of following block velocity and determination of override points) can be used to calculate the velocity profiles and is generally advisable because the preset machine data for these functions already takes the widest range of override-dependent velocity limits into account.

Note

If neither of the procedures has been activated, the setpoint velocity is always applied in the current block.

Note

Plannable velocity limits restrict override-specific velocity limits.

Relief factor with block cycle problems

Block cycle problems are encountered in cases where the traversing distances of the NC blocks to be processed are so short that the LookAhead function has to reduce the machine velocity to provide enough time for block processing. In this situation, constant braking and acceleration of path motion may occur.

Velocity fluctuations of this type can be dampened by specifying a relief factor:

MD20450 $MC_LOOKAH_RELIEVE_BLOCK_CYCLE (relieving factor for the block cycle time)

Supplementary conditions

Axis-specific feed stop/axis disable

Axis-specific feed stop and axis-specific axis disable are ignored by LookAhead.

If an axis is to be interpolated that should on the other hand be made stationary by axis-specific feed stop or axis disable, LookAhead does not stop path motion before the block in question but decelerates **in** the block itself.

If this response is not wanted, an **axis**-specific feed stop can be transferred to a **channel** via the PLC to stop the path immediately (see also Section "Clamping monitoring (Page 93)").

3.3.4.2 Free-form surface mode: Extension function

Function

The "Free-form surface mode: Extension function" is an extension of the LookAhead standard functionality and is used to calculate the path velocity profile during free-form surface processing (see also Section "Free-form surface mode: Basic functions (Page 222)").

Its use optimizes the continuous-path mode as follows:

- Symmetry between the acceleration and deceleration profiles
- Uniform acceleration process, even with changing jerk or acceleration limits
- Uniform acceleration process of target velocity profiles, irrespective of the degree to which they can or cannot be started with the specified dynamic response limit
- LookAhead braking to lower setpoint velocities

Uniformity and compliance with the dynamic response limit guarantee that the setpoint velocity profiles are smoothed to a homogeneous velocity profile on the part. This serves to minimize the effect of following errors on the quality of the surface.

Therefore, the function offers the following advantages:

- Greater uniformity in the surface of the workpiece
- Lower machine load

Applications

The "Free-form surface mode: Extension function" is used to process workpieces which primarily comprise free-form surfaces.

Note

As better results are not achieved for standard processing applications, standard LookAhead functionality should be used in these cases.

Requirements

- The function is only effective:
  - In AUTOMATIC
  - In acceleration mode: acceleration with jerk limit (SOFT)
- Activation is only possible if the requisite memory is configured:

  MD28533 $MC_MM_LOOKAH_FFORM_UNITS = <value>

  Sensible value assignment depends upon the part program, the block lengths, the axial dynamic response, as well as upon an active kinematic transformation.

  The following setting applies as a guideline for processing free-form surfaces: 18

  Note

  Due to the additional storage requirements, MD28533 should only be set for the channels in which free-form surfaces are being processed.

Activation/deactivation

The function can be switched on or off independently for every dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 219)"):

MD20443 $MC_LOOKAH_FFORM[<n>]= <value>

Index <n>	Dynamic response mode	<value>	Free-form surface mode: Extension function
0	Standard dynamic response settings (DYNNORM)	0	Off
		1	On
1	Positioning mode, tapping (DYNPOS)	0	Off
		1	On
2	Roughing (DYNROUGH)	0	Off
		1	On
3	Finishing (DYNSEMIFIN)	0	Off
		1	On
4	Smooth finishing (DYNFINISH)	0	Off
		1	On

The "Free-form surface mode: Extension function" is typically only active if the "Free-form surface mode: Basic functions" are also active. Therefore, the settings in MD20443 $MC_LOOKAH_FFORM[<n>] should correspond to the settings in MD20606 $MC_PREPDYN_SMOOTHING_ON[<n>].

The standard LookAhead functionality is active in the dynamic response modes in which the "Free-form surface mode: Extension function" is switched off.

Programming

Generally speaking, the "Free-form surface mode: Extension function" becomes effective as a result of a change in the dynamic response mode in the part program.

Example

The following parameters are assumed:

MD20443 $MC_LOOKAH_FFORM[0] = 0

MD20443 $MC_LOOKAH_FFORM[1] = 0

MD20443 $MC_LOOKAH_FFORM[2] = 1

MD20443 $MC_LOOKAH_FFORM[3] = 1

MD20443 $MC_LOOKAH_FFORM[4] = 1

```
Program code                                    Comment
N10 DYNPOS                                      ; Switch on the DYNPOS dynamic
                                                  response mode. Standard LookAhead
                                                  functionality is active in the
                                                  DYNPOS dynamic response mode.
...
N100 G17 G54 F10000
N101 DYNFINISH                                  ; Switch on the DYNFINISH dynamic
                                                  response mode. The "Free-form
                                                  surface mode: Extension functions"
                                                  are active in the DYNFINISH dynamic
                                                  response mode.
N102 SOFT G642
N103 X-0.274 Y149.679 Z100.000 G0
N104 COMPCAD
...
N1009 Z4.994 G01
N10010 X.520 Y149.679 Z5.000
N10011 X10.841 Y149.679 Z5.000
N10012 X11.635 Y149.679 Z5.010
N10013 X12.032 Y149.679 Z5.031
M30
```

Note

When switching between the standard LookAhead functionality and the "Free-form surface mode: Extension function" or vice versa, continuous-path mode is interrupted by an interpolator stop.

Supplementary conditions

Automatic function switchover

Applying the following functions when the "Free-form surface mode: Extension function" leads to an automatic switchover to standard LookAhead functionality:

- Thread cutting/tapping (G33, G34, G35, G331, G332, G63)
- Path master-value coupling
- Punching, nibbling
- Cartesian PTP travel

The "Free-form surface mode: Extension function" is then switched on again automatically.

Using the commands of G function group 15 (feed types)

It is not advisable to use the following feed types when the "Free-form surface mode: Extension function" is active:

- Feedrate per revolution (G95, G96, G97, etc.)
- Inverse-time feedrate (G93)

Time response during feedrate overrides

$ Feedrate overrides (via a machine control panel, $AC_OVR, ...) can extend the traverse time over standard LookAhead functionality considerably.

Rapid traverse motion (G0)

G0 blocks which are interspersed during free-form surface processing do not switch the LookAhead functionality over (from the "Free-form surface mode: Extension function" to standard LookAhead functionality or vice versa).
This means that even though the standard dynamic response setting (DYNNORM) is effective with G0, the standard LookAhead functionality which is preset for DYNNORM (→ MD20443 $MC_LOOKAH_FFORM[0]) does not automatically become effective as well as a result.

By retaining the LookAhead functionality which is currently active, a more homogeneous velocity profile is achieved, particularly since G0 and polynomial blocks are usually smoothed and connected by rounding.

Number of NC blocks in the IPO buffer

It is generally advisable to significantly increase the configured number of NC blocks in the interpolation buffer when using the "Free-form surface mode: Extension function":

MD28060 $MC_MM_IPO_BUFFER_SIZE > 100

If the block memory capacity is too low, this may diminish the uniformity of the path-velocity profile.

3.4 Dynamic adaptations

3.4.1 Smoothing of the path velocity

Introduction

The velocity control function utilizes the specified axial dynamic response. If the programmed feedrate cannot be achieved, the path velocity is brought to the parameterized axial limit values and the limit values of the path (velocity, acceleration, jerk). This can lead to repeated braking and acceleration on the path.

If a short acceleration takes place during a machining function with high path velocity, and is thus followed almost immediately by braking, the reduction in the machining time is only minimal. Acceleration of this kind can, however, have undesirable effects if, for example, it results in machine resonance.

In some applications in mold making, especially in the case of high-speed cutting, it is desirable to achieve a constant path velocity. In these cases, it can therefore be reasonable to sacrifice transient acceleration processes in favor of a smoother tool path velocity.

Function

If the "smoothing the path velocity" function is active, a smoothing factor, which determines the maximum permissible productivity loss, takes effect with a view to achieving smoother path velocity control: Acceleration processes which contribute less than this factor to a shorter program runtime are not performed. Account is only taken of acceleration processes whose frequencies lie above the configurable limit frequencies of of the axes involved.

Benefits:

- Avoidance of excitations of possible machine resonance due to continuous, transient braking and acceleration processes (in the area of less IPO cycles).
- Avoidance of constantly varying cutting rates due to acceleration which brings no significant shortening of the program running time.

Note

The smoothing of the path velocity does not lead to contour errors.

Variations in axis velocity due to curvatures in the contour at constant path velocity may continue to occur and are not reduced with this function.

Variations in path velocity due to the input of a new feedrate are not changed either. This remains the responsibility of the programmer of the subprogram.

Requirements

- The smoothing of the path velocity is only effective in continuous-path mode with LookAhead over multiple blocks with SOFT and BRISK. Smoothing is **not** effective with G0.
- The controller's cycle times must be configured in such a way that preprocessing can prepare sufficient blocks to enable an acceleration process to be analyzed.

Activation/deactivation

The "smoothing of the path velocity" function is activated/deactivated with the machine data:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)

Value	Meaning
0.0	Smoothing of the path velocity **not** active (default)
> 0	Smoothing of the path velocity active

A change in the MD setting is only made effective through NEW CONF.

Parameterization

Smoothing factor

The smoothing factor is set via the channel-specific machine data:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)

The percentage value defines how much longer a processing step without accelerations/decelerations may be than the corresponding step with accelerations/decelerations.

This would be a "worst-case" value, if all accelerations within the part program, except the initial approach motion, were smoothed. The actual extension will always be smaller, and may even be 0, if the criterion is not met by any of the accelerations. Values between 50 and 100% may also be entered without significantly increasing the machining time.

Consideration of the programmed feed

The path velocity can be smoothed with or without taking the programmed feedrate into consideration. The selection is made via the machine data:

MD20462 $MC_LOOKAH_SMOOTH_WITH_FEED (path smoothing with programmed feedrate)

Value	Meaning
0	Programmed feedrate is **not** taken into consideration.
1	Programmed feedrate is considered (default setting).

When considering the programmed feedrate, the specified smoothing factor (see MD20460) is maintained better when the override is 100%.

Axis-specific limit frequencies

The axis-specific limit frequencies are defined via the machine data:

MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead)

Acceleration and deceleration processes, which run with a high frequency, are smoothed depending upon the parameterization of the following machine data or else are reduced in dynamics:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)

MD20465 $MC_ADAPT_PATH_DYNAMIC (adaptation of the dynamic path response)

For further information on MD20465, see Section "Adaptation of the dynamic path response (Page 210)".

Note

If vibrations are generated in the mechanical system of an axis and if the corresponding frequency is known, MD32440 should be set to a value smaller than this frequency.

The needed resonance frequencies can be calculated using the built-in measuring functions.

Mode of operation

The minimum value for MD32440 is calculated as f_{path} on the basis of the axes involved in the path. For the smoothing only those acceleration processes are taken into consideration, in which the start and the end velocity of this motion are reached within the time given below:

$t = t_2 - t_1 = 2 / f_{path}$

These acceleration processes are dispensed with if the resulting extension in the processing time does not exceed the limit specified in excess of the smoothing factor (MD20460).

Example

The following parameters are assumed:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 10%

MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 Hz

MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 20 Hz

MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 10 Hz

The path involves the three axes X = AX1, Y = AX2, Z = AX3.

The minimum value of MD32440 for these three axes is thus 10 Hz. This means that any acceleration, which is completed within a period of $t_2 - t_1$ = 2/10 Hz = 200 ms, is examined. The time t_2 is the time it takes to reach velocity v_1 again following an acceleration process starting from velocity v_1. The extending of the execution time is also only considered within this range.

If the time $t_2 - t_1$ is greater than 200 ms or if the additional program execution time $t_3 - t_2$ is more than 10% (= MD20460) of $t_2 - t_1$, the following time characteristic applies:

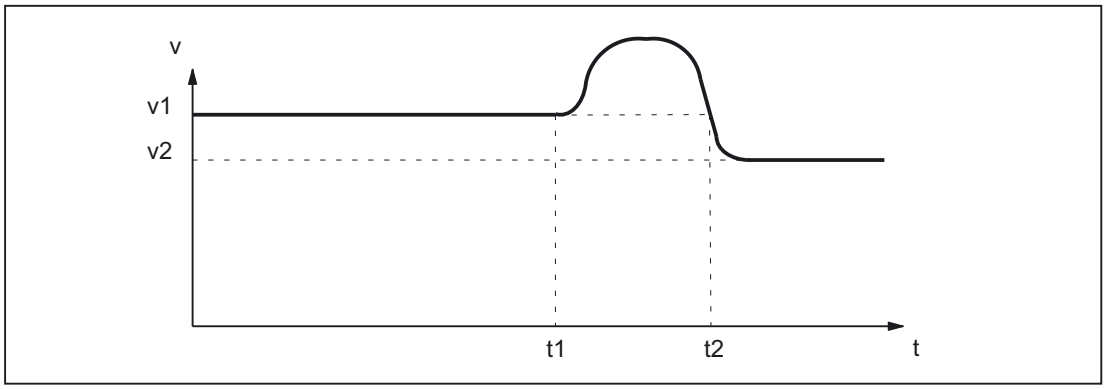


Figure 3-7 Characteristic of time-optimum path velocity (without smoothing)

If, however, the time $t_2 - t_1$ is less than 200 ms or if the additional program execution time $t_3 - t_2$ is no more than 10% of $t_2 - t_1$, the following time characteristic applies:

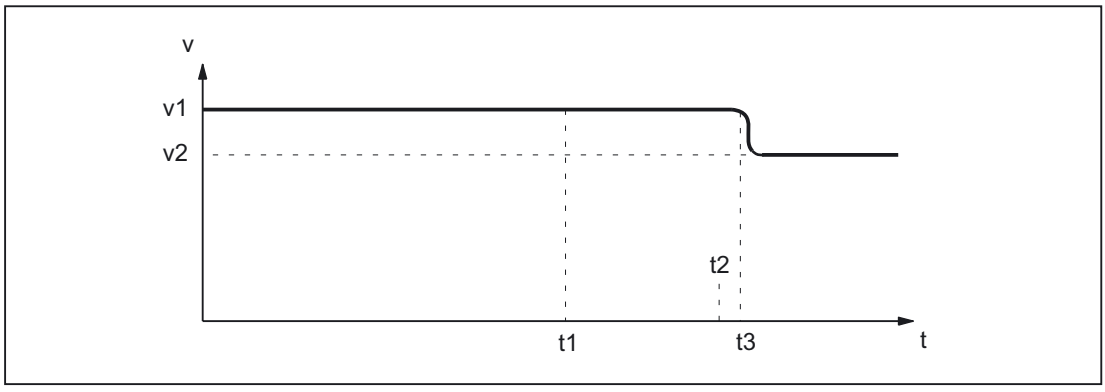


Figure 3-8 Characteristic of the smoothed path velocity

3.4.2 Adaptation of the dynamic path response

Function

Highly dynamic acceleration and deceleration processes during machining can cause excitation of mechanical vibrations of machine elements and consequently a reduction of the surface quality of the workpiece.

The dynamic response of the acceleration and deceleration processes can be adapted to the machine conditions using the "adaptation of the dynamic path response" function.

Note

The "adaptation of the dynamic path response" function only concerns the resulting path and not the deceleration and acceleration processes of the individual axes involved in the path. For this reason, critical deceleration and acceleration processes of the axes with respect to the excitation of mechanical vibrations can occur due to discontinuous contour profiles or kinematic transformations, even with a constant path velocity profile.

Effectiveness

The "adaptation of the dynamic path response" function is only effective during path motions:

- Continuous-path mode (G64, G64x)

  In continuous-path mode, the optimal effect of the dynamic response adaptation is attained with an active 100% override. Considerable deviations from this value or functions that cause the path axes to decelerate (e.g. auxiliary function outputs to the PLC) greatly reduce the desired action.
- Exact stop (G60)

In addition, the "adaptation of the dynamic path response" function is **not** active during path motions:

- Programmed rapid traverse (G0)
- Changes in the override value
- Stop requests during motion (e.g. NC Stop, NC Reset)
- "Velocity-dependent path acceleration" function (DRIVE) is active

Activation/deactivation

The function is activated/deactivated with the machine data:

MD20465 $MC_ADAPT_PATH_DYNAMIC (adaptation of the dynamic path response)

Value	Meaning
= 1.0	Dynamic adaptation **not** active (default setting)
> 1.0	Dynamic adaptation active

When activation takes place, the **"smoothing the path velocity" function is always activated** internally in continuous-path mode as well (see Section "Smoothing of the path velocity (Page 206)").

If the smoothing factor (MD20460 $MC_LOOKAH_SMOOTH_FACTOR) is set to 0% (= function deactivated; default!), a smoothing factor of 100% is used as a substitute. For a smoothing factor other than 0%, the set value takes effect.

Parameterization

Adaptation factor of the dynamic path response

Via the adaptation factor of the dynamic path response, temporary changes in the path velocity are executed with smaller dynamic response limit values.

The adaptation factor is to be set on a channel-specific basis:

- For traversing motions with acceleration without jerk limitation (**BRISK**) via:

  MD20465 $MC_ADAPT_PATH_DYNAMIC[**0**]

  → The adaptation factor acts on the acceleration.

- For traversing motions with acceleration with jerk limitation (**SOFT**) via:

  MD20465 $MC_ADAPT_PATH_DYNAMIC[**1**]

  → The adaptation factor acts on the jerk.

Axis-specific limit frequencies

The dynamic response limiting should only be active during deceleration and acceleration processes that trigger mechanical vibrations larger than a specific limiting frequency, thus causing excitation of machine resonances.

This limit frequency from which the dynamic response limiting activates, is specified on an axis-specific basis via the machine data:

MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead)

For further information, see Section "Smoothing of the path velocity (Page 206)".

Mode of operation

During processing and via all the axes involved in the path, the controller cyclically establishes the minimum of all the limit frequencies to be the limit frequency (f) for the adaptation of the dynamic response and calculates the relevant time window (t_{adapt}) from this:

$t_{adapt} = 1 / f$

The size of the relevant time window t_{adapt} determines the further behavior:

1. The time needed to change the velocity is less than t_{adapt}:

   The acceleration rates are reduced by a factor > 1 and ≤ the value written in machine data:

   MD20465 ADAPT_PATH_DYNAMIC (adaptation of the path dynamics)

   The reduction in acceleration rate increases the time taken to change the velocity.

   The following cases are different:

   – The acceleration rate is reduced with a value less than MD20465 so that the process lasts for t_{adapt} [s]. The permitted reduction does not need to be fully utilized.
   – The acceleration time is reduced with the value written in MD20465. The process lasts less than t_{adapt} despite the reduced acceleration. The permitted reduction was fully utilized.

2. The time needed to change the velocity is greater than t_{adapt}:

   No dynamic response adaptation is required.

Example

The following example is intended to show the effect of the "adaptation of the dynamic path response" function on traversing motions with acceleration and without jerk limitation (BRISK).

The following parameters are assumed:

MD20465 $MC_ADAPT_PATH_DYNAMIC[0] = 1.5
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 1.0
MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 Hz T_{AX1} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 10 Hz T_{AX2} = 1/10 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20 Hz T_{AX3} = 1/20 Hz = 50 ms

Note

To illustrate the effect of dynamic response adaptation, the value for the smoothing factor (MD20460) is set to "1", whereby the "smoothing of the path velocity" function is practically deactivated.

The path involves the three axes X = AX1, Y = AX2, Z = AX3.

For path motions in which axis AX2 is involved, all deceleration and acceleration processes that would last less than T_{AX2} are adapted.

If only axes AX1 and/or AX3 are involved in path motions, all deceleration and acceleration processes that would last less than $T_{AX1} = T_{AX3}$ are adapted.

The relevant time window is marked $t_{adapt...}$ in the figures below.

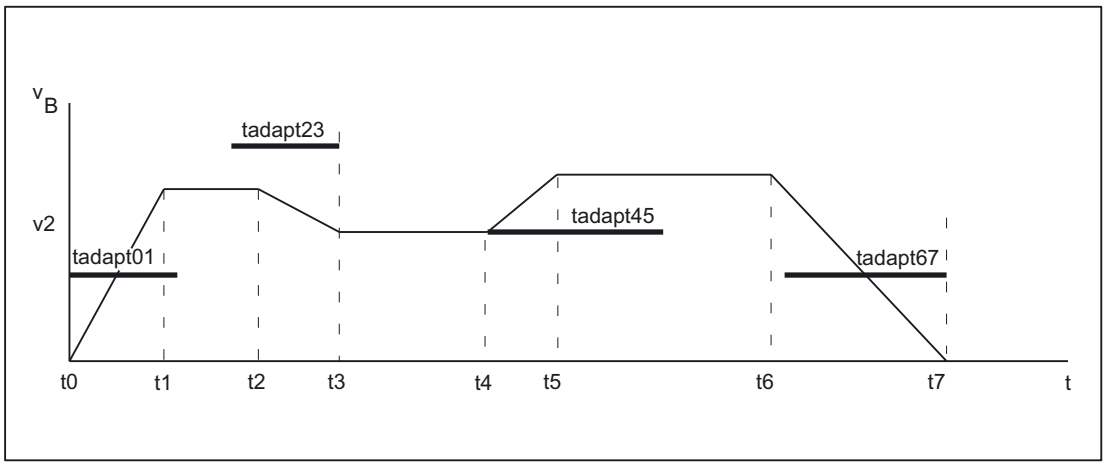


Figure 3-9 Path velocity profile optimized for time without smoothing or dynamic adaptation response

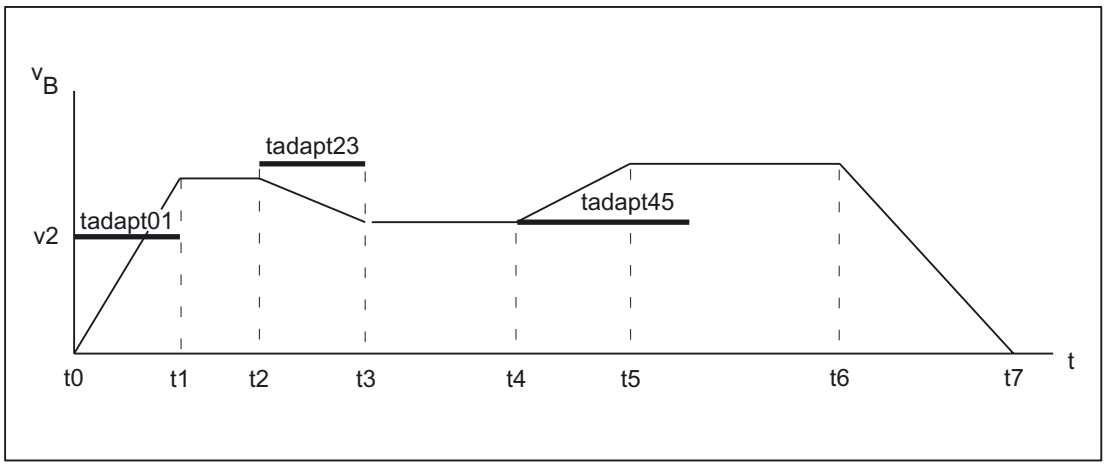


Figure 3-10 Path velocity profile with adaptation of dynamic path response

Intervals t_0 - t_1 and t_2 - t_3:	The acceleration process between t_0 - t_1 and the deceleration process between t_2 - t_3 are extended in terms of time to $t_{adapt01}$ or $t_{adapt23}$ as a result of the acceleration being adapted.
Interval t_4 - t_5:	The acceleration process between t_4 - t_5 is executed with an acceleration reduced by the maximum adaptation factor of 1.5. However, the acceleration process is completed before time $t_{adapt45}$.
Interval t_6 - t_7:	The deceleration process between t_6 - t_7 remains unchanged as it lasts longer than $t_{adapt67}$.

3.4.3 Determination of the dynamic response limiting values

In addition to determining the natural frequency of the path axes for assigning parameters to the axis-specific limit frequencies (MD32440 $MA_LOOKAH_FREQUENCY), the implementation of the "adaptation of the dynamic path response" function also requires dynamic response limits to be determined for velocity, acceleration and jerk.

Procedure

The determination of the dynamic response limits for the traversing of path axes by means of acceleration with jerk limiting (SOFT) is described below. This procedure can be applied by analogy to the case of acceleration without jerk limiting (BRISK).

1. Deactivate the "adaptation of the dynamic path response" function:

   MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 1

2. Observe the positioning behavior of each path axis at different traversing velocities. When doing so, set the jerk such that the desired positioning tolerance is maintained.

   Note

   The higher the traversing velocity from which the positioning process is started, the higher in general the jerk can be set.

3. Use the maximum permissible jerk determined for the least critical traversing velocity:

   MD32431 $MA_MAX_AX_JERK (maximum jerk)

4. Determine the F_{APD} factor for all of the path axes using:

   F_{APD} = (largest determined jerk) / (smallest determined jerk)

   Note

   The smallest determined jerk is the value for the jerk during the most critical traversing velocity.

5. Enter the largest F_{APD} factor that was determined via all the path axes as the value for the adaptation factor for the path dynamic response:

   MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = F_{APD}

3.4.4 Interaction between the "smoothing of the path velocity" and "adaptation of the path dynamic response" functions

The following examples serve to illustrate the interaction between the "smoothing of the path velocity" and "adaptation of the path dynamic response" functions in continuous-path mode.

Example 1

Acceleration mode: BRISK

The path involves the 3 axes X = AX1, Y = AX2, Z = AX3.

The following parameters are assumed:

MD20465 $MC_ADAPT_PATH_DYNAMIC[0] = 3
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 80.0
MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 T_{AX1} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 20 T_{AX2} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20 T_{AX3} = 1/20 Hz = 50 ms

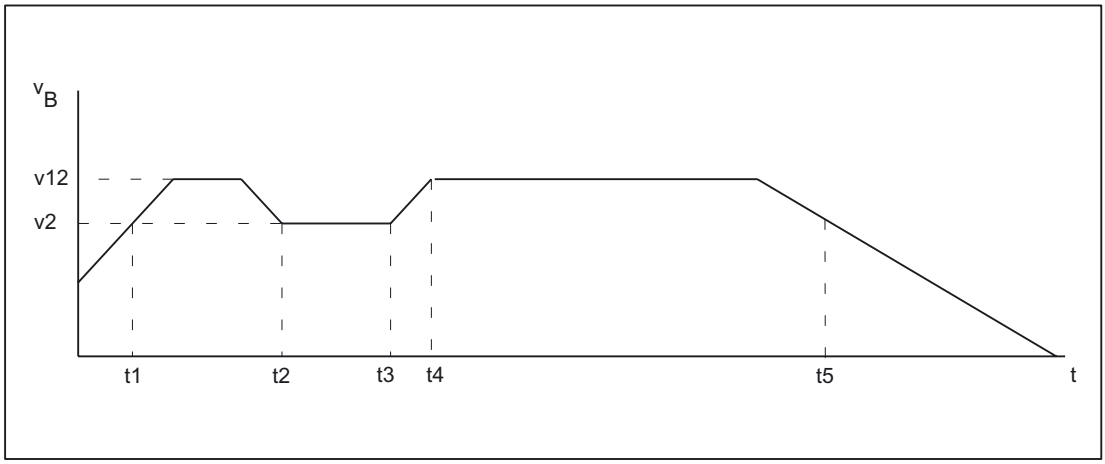


Figure 3-11 Path velocity profile optimized for time without smoothing or dynamic adaptation response

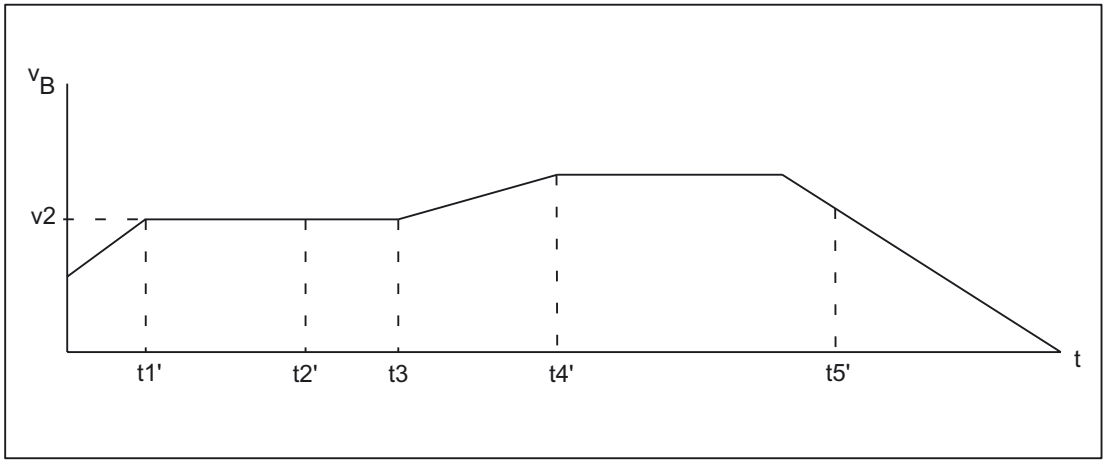


Figure 3-12 Path velocity profile with smoothing of the path velocity and adaptation of dynamic path response

Effects of smoothing on path velocity:

Interval t_1 - t_2:	The acceleration and deceleration process between t_1 and t_2 does not take place because the lengthening of the machining time without the acceleration process to v_{12} is less than the resulting time if a smoothing factor of 80 % is applied.
Interval t_3 - t_5:	The acceleration and braking profile between t_3 and t_5 does not fulfill this condition or takes longer than the parameterized smoothing time T_{Axn} = 2/20 Hz = 100 ms.

Effects of the dynamic response adaptation:

Interval t_3 - t_4:	The acceleration process between t_3 and t_4 is shorter than MIN(T_{AXn}) = 1/20 Hz = 50 ms and is, therefore, executed with an acceleration reduced by an adaptation factor of 3.
Interval up to t_1:	The acceleration up to t_1 left over after path smoothing is stretched to the time period up to t_1' by the dynamic response adaptation.

Note

The example shows that those acceleration or deceleration processes that are not eliminated by the smoothing of the path velocity can be subsequently optimized by adapting the dynamic path response. For this reason, both functions should always be activated, if possible.

Example 2

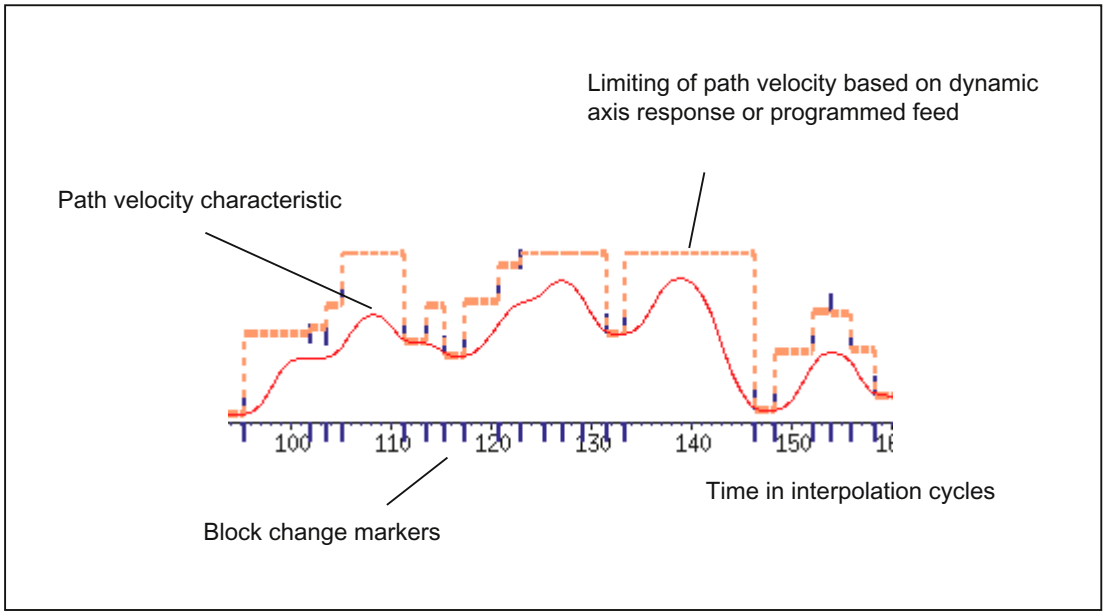
Acceleration mode: SOFT

The path involves the 3 axes X = AX1, Y = AX2, Z = AX3.

The following parameters are assumed:

MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 1
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 0.0
MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 10 T_{AX1} = 1/20 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 10 T_{AX2} = 1/20 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20 T_{AX3} = 1/20 Hz = 50 ms

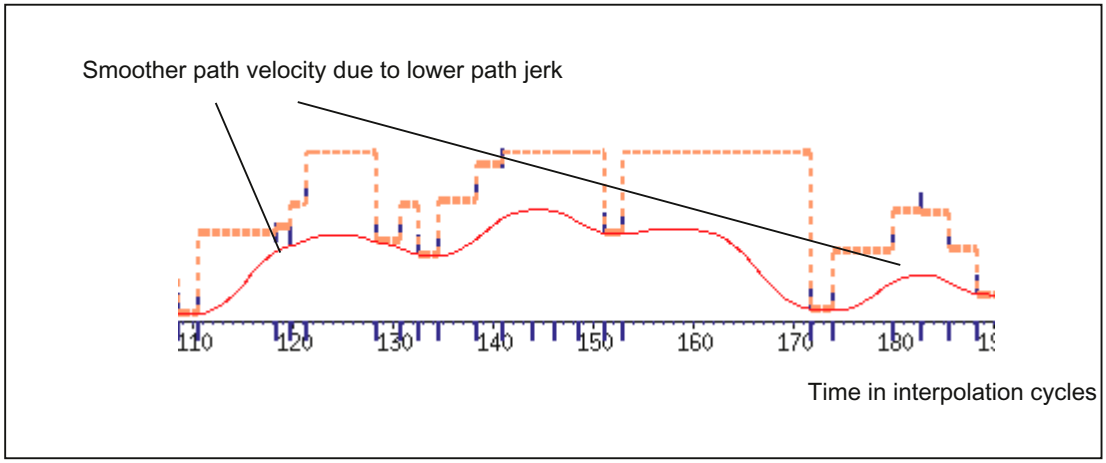
This leads to a path velocity profile which is optimized in terms of time without smoothing the path velocity or adapting the dynamic path response:



The parameter assignment is changed as follows:

MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 4
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 1.0

This results in a path velocity profile with adaptation of the dynamic path response and with minimum, and thus virtually deactivated, smoothing of the path velocity:

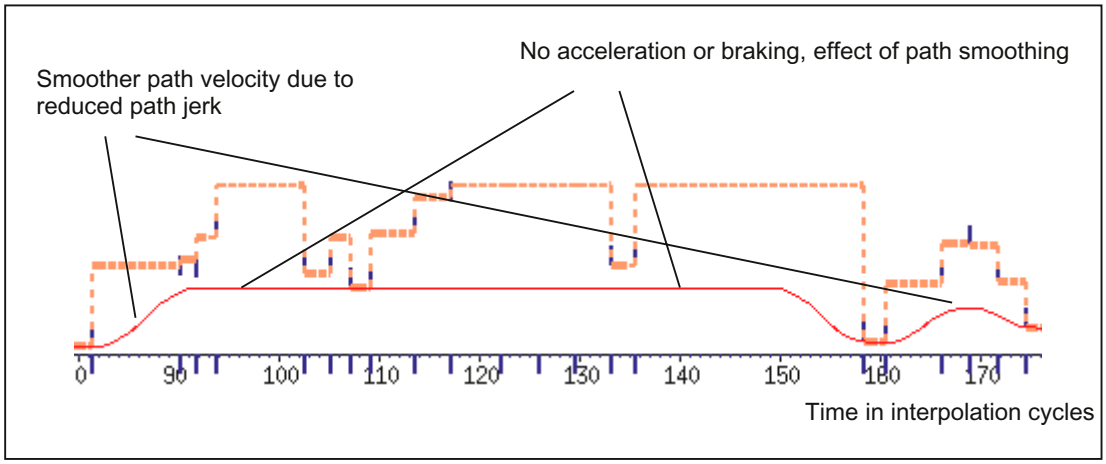


The smoothing factor is set to 0% instead of 1% (in accordance with the default!):

MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 0.0

A smoothing factor of 100% comes into effect with this parameter assignment.

This gives rise to a path velocity profile with smoothing of the path velocity and adaptation of dynamic path response:



3.4.5 Dynamic response mode for path interpolation

Function

Technology-specific, dynamic response settings can be saved in machine data and can be activated in the part program via the commands from G function group 59 (dynamic response mode for path interpolation).

Command	Activates the dynamic response settings for:
DYNNORM	Standard dynamic response settings
DYNPOS	Positioning mode, tapping
DYNROUGH	Roughing
DYNSEMIFIN	Finishing
DYNFINISH	Smooth finishing

Note

The dynamic response of the path axes alone is determined by the commands from G function group 59 (dynamic response mode for path interpolation). They have no effect on:

- Positioning axes
- PLC axes
- Command axes
- Motions based on axis coupling
- Overlaid motions with handwheel
- JOG motions
- Reference point approach (G74)
- Fixed-point approach (G75)
- Rapid traverse motion (G0)

The standard dynamic response setting (DYNNORM) always takes effect for these axis motions.

Application

By switching the dynamic response settings, roughing a workpiece can be optimized in terms of time and smoothing it can be optimized in terms of the surface, for example.

Parameterization

Axial machine data

Number	Identifier: $MA_	Description
Axis-specific dynamic response settings		
MD32300	MAX_AX_ACCEL[<n>]	Axis acceleration
MD32431	MAX_AX_JERK[<n>]	Max. axial jerk for path motion
MD32432	PATH_TRANS_JERK_LIM[<n>]	Max. axial jerk at the block transition in continuous-path mode
MD32310	MAX_ACCEL_OVL_FACTOR[<n>]	Overload factor for axial velocity jumps
MD32433	SOFT_ACCEL_FACTOR[<n>]	Scaling of acceleration limitation for SOFT
Path-related dynamic response settings		
MD20600	MAX_PATH_JERK[<n>]	Path-related maximum jerk

Channel-specific machine data

Number	Identifier: $MC_	Description
Path-related dynamic response settings		
MD20602	CURV_EFFECT_ON_PATH_ACCEL[<n>]	Influence of path curvature on the path acceleration
MD20603	CURV_EFFECT_ON_PATH_JERK[<n>]	Influence of path curvature on the path jerk

Range of values for index n:

Index <n>	Meaning
0	Value for DYNNORM
1	Value for DYNPOS
2	Value for DYNROUGH
3	Value for DYNSEMIFIN
4	Value for DYNFINISH

NOTICE

Writing the machine data **without an index** places the same value in all field elements of the relevant machine data.

Reading the machine data **without an index** always supplies the value of the field with index 0.

Suppressing G commands

It is recommended that G commands from G function group 59 (dynamic response mode for path interpolation) which are not intended for use should be suppressed via the following machine data:

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[<n>] (list of reconfigured NC commands)

When using a suppressed G command, an alarm is displayed in order to ensure that no non-parameterized machine data takes effect.

Example

The G commands `DYNPOS` and `DYNSEMIFIN` can be suppressed with the following settings:

- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[0]="DYNPOS"
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[1]=" "
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[2]="DYNSEMIFIN"
- MD10712 $MN_ NC_USER_CODE_CONF_NAME_TAB[3]=" "

References

You can find further information about programming the G commands from G function group 59 (dynamic response mode for path interpolation) in:
References:
Programming Manual, Fundamentals; Section: Path action

3.4.6 Free-form surface mode: Basic functions

Introduction

In applications in tool and mold making, it is important that the surfaces on the workpiece are as uniform as possible. This requirement is generally more important than the precision of the surface of the workpiece.

Workpiece surfaces which lack uniformity can be attributable to the following causes, for example:

- The part program for manufacturing the workpiece contains a non-uniform geometry. This, most notably, affects the profile of the curvature and torsion.

  Note

  The curvature k of a contour is the inverse of radius r of the nestling circle in a contour point (k = 1/r). The torsion is the change in curvature (first derivative).

  As a result of the lack of uniformity in geometry, the machine's dynamic response limits are reached during processing of the part program, and needless deceleration and acceleration processes occur. Depending on the extent of the effective over-travel of the axes, this leads to different deviations in contours.

- Needless deceleration and acceleration processes can trigger machine vibrations which result in unwanted marks on the workpiece.

There are various options available for eliminating these causes:

- The part programs generated by the CAD/CAM system contain a very uniform curvature and torsion profile, preventing needless reductions in path velocity.
- The maximum path velocity is determined in such a way that unwanted geometric fluctuations in the curvature and torsion profile have no effect.

Function

"Free-form surface mode: Basic functions" can be used to make the definition of path velocity limits insensitive to small geometric fluctuations in curvature and torsion without exceeding the machine's dynamic limits in terms of the acceleration and jerk of the axes.

This has the following advantages:

- Greater uniformity in the profile of the path velocity
- Greater uniformity in the surface of the workpiece
- Reduction in the processing time (if the dynamic response of the machine permits it)

Applications

The function is used to process workpieces which primarily comprise free-form surfaces.

Requirements

The function can only be activated if the requisite memory capacity is reserved during memory configuration:

MD28610 $MC_MM_PREPDYN_BLOCKS = **10**

The value entered prescribes the number of blocks which have to be taken into consideration in the determination of the path velocity (velocity preparation).
A sensible value is "10".

If MD28610 has a value of "0", only the motions of the axes in a particular block are taken into consideration when determining the maximum velocity of the path for that block. If the geometry of neighboring blocks is also taken into consideration when determining the velocity of the path (value > 0), a more uniform profile in path velocity is achieved.

Activation/deactivation

The function can be switched on or off independently for every dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 219)"):

MD20606 $MC_PREPDYN_SMOOTHING_ON[<n>] = <value>

Index <n>	Dynamic response mode	<value>	Free-form surface mode: Basic functions
0	Standard dynamic response settings (DYNNORM)	0	Off
		1	On
1	Positioning mode, tapping (DYNPOS)	0	Off
		1	On
2	Roughing (DYNROUGH)	0	Off
		1	On
3	Finishing (DYNSEMIFIN)	0	Off
		1	On
4	Smooth finishing (DYNFINISH)	0	Off
		1	On

Note

Due to the additional storage requirements, the function should only be activated in the relevant processing channels.

Parameterization

Change in the contour sampling factor

The secant error which occurs during the interpolation of curved contours is dependent on the following factors:

- Curvature
- Interpolation cycle (display in the MD10071 $MN_IPO_CYCLE_TIME)
- Velocity with which the relevant contour is traversed

The maximum possible secant error is defined for each axis in the machine data:

MD33100 $MA_COMPRESS_POS_TOL (maximum tolerance with compression)

If the set interpolation cycle is not sufficiently small, the max. path velocity may be reduced in the case of contours with greater curvature. This is necessary for ensuring that the surface of the workpiece is also produced with an adequate degree of precision in this case.

By changing the contour sampling factor, the time interval with which a curved contour is sampled in the interpolator (contour sampling time) can be set at variance with the interpolation cycle. A contour sampling time which is shorter than the interpolation cycle can prevent a reduction in path velocity in the case of contours with greater curvature.

The contour sampling factor is set with the machine data:

MD10682 $MN_CONTOUR_SAMPLING_FACTOR

The effective contour sampling time is calculated as follows:

$T_s = f * T_1$

where: T_s = Effective contour sampling time
T_1 = Interpolation cycle time
f = Contour sampling factor (value from MD10682)

The default contour sampling factor is "1", i.e. the contour sampling time equals the interpolation cycle.

The contour sampling factor can be both greater or less than "1".

If a value less than "1" is set, monitoring of contour sampling precision is disabled.

The set sampling time must not be below the configured minimum contour sampling time:

MD10680 $MN_MIN_CONTOUR_SAMPLING_TIME

Note

MD10680 is specifically set for every controller model and cannot be changed.

Programming

Depending on the setting in machine data MD20606 $MC_PREPDYN_SMOOTHING_ON, "Free-form surface mode: Basic functions" can be switched on and off in the part program by changing the active dynamic response mode.

Example:

By assigning the parameters MD20606 $MC_PREPDYN_SMOOTHING_ON[2-4] = 1 and MD20606 $MC_PREPDYN_SMOOTHING_ON[0-1] = 0, the function can be switched on via the commands `DYNROUGH`, `DYNSEMIFIN`, and `DYNFINISH` and switched off via the commands `DYNNORM` and `DYNPOS`.

See also

Rounding of tangential block transitions (G645) (Page 193)

Velocity-dependent jerk adaptation (axis-specific) (Page 273)

Free-form surface mode: Extension function (Page 201)

3.5 Compressor functions

3.5.1 NC block compression

Function

COMPON, COMPCURV

The compressor functions COMPON and COMPCURV generate one polynomial block from up to ten consecutive linear blocks of the form: "G01 X... Y... Z... F...". The polynomial blocks of the compressor functions have the following properties:

- COMPON: Continuous velocity block transitions.
- COMPCURV: Continuous velocity and acceleration block transitions

COMPCAD

The compressor function COMPCAD can generate one polynomial block from theoretically any number of linear and circular blocks. The polynomial blocks have constant velocity and acceleration at the block transitions. Corners that are desirable are identified as such and taken into account.

The maximum tolerable deviation of the calculated path to the programmed points can be specified using machine data for all compressor functions. In contrast to COMPON and COMPCURV, the specified tolerances are not used in different directions in neighboring paths with COMPCAD. In fact, COMPCAD attempts to achieve - under similar conditions - also similar deviations from the programmed points.

The common objective of compressor functions is to optimize the surface quality and machining speed by achieving continuous block transitions and increasing the path length for each block.

COMPCAD is very CPU time and memory-intensive. It is recommended that COMPCAD is only used there where surface improvements were not successful using measures in the CAD/CAM program.

General

- The position data in the blocks to be compressed can be realized as required, e.g. X100, X=AC(100), X=R1*(R2+R3)
- The compression operation is then interrupted by every other command, e.g. auxiliary function output, in and between the blocks to be compressed.

Availability

For SINUMERIK 828D, NC block compression is only available for the milling versions.

Orientation transformation (TRAORI)

When orientation transformation (TRAORI) is active, and under certain conditions, the compressor functions COMPON, COMPCURV and COMPCAD can also compress motion blocks for tool orientation and tool rotation. A detailed description can be found in:

References:
Function Manual Special Functions; 3- to 5-Axis Transformation (F2),
Section: "Compression of the orientation (COMPON, COMPCURV and COMPCAD)"

Parameterization

The following machine and setting data must be set for the parameterization of the NC block compression:

Channel-specific machine data

Number	Identifier $MC_	Meaning
MD20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of NC block for compression
MD20172	COMPRESS_VELO_TOL	Maximum permissible deviation from path feed for compression
MD20482	COMPRESSOR_MODE	Setting the mode of operation of the compressor See also: **References**: Function Manual, Special Functions; Multi-Axis Transformations (F2), Section: "Orientation" > "Compression of the orientation"

Channel-specific setting data

Number	Identifier $SC_	Meaning
SD42470	CRIT_SPLINE_ANGLE	Corner limit angle for COMPCAD
SD42475	COMPRESS_CONTUR_TOL	Maximum permissible contour deviation with compression

Note

Corner limit angle and compressor function COMPCAD

The corner limit angle for COMPCAD set via the setting data SD42470 $SC_CRIT_SPLINE_ANGLE is only used as an approximate measure for corner detection. By evaluating the plausibility, the compressor can also identify flatter block transitions as corners and larger angles as outliers.

Axial machine data

Number	Identifier $MA_	Meaning
MD33100	COMPRESS_POS_TOL	Maximum permissible path deviation with compression

Recommended settings for retroactive machine data

When using the compressor function, the following settings are recommended for the retroactive machine data on the compressor function:

Number	Identifier	Recommended value
MD18360	$MN_MM_EXT_PROG_BUFFER_SIZE (FIFO buffer size for execution from external source)	100
MD20490	$MC_IGNORE_OVL_FACTOR_FOR_ADIS (G641/G642 irrespective of the overload factor)	1
MD28520	$MC_MM_MAX_AXISPOLY_PER_BLOCK (maximum number of axis polynomials per block)	3
MD28530	$MC_MM_PATH_VELO_SEGMENTS (number of memory elements for limiting the path velocity)	5
MD28540	$MC_MM_ARCLENGTH_SEGMENTS (number of memory elements for displaying the arc length function)	10
MD28060	$MC_MM_IPO_BUFFER_SIZE (number of NC blocks for the block preparation)	100
MD28070	$MC_MM_NUM_BLOCKS_IN_PREP (number of blocks for the block preparation)	60
MD32310	$MA_MAX_ACCEL_OVL_FACTOR (overload factor for axial velocity jumps)	<Value for G64 operation>

Programming

Switch on

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

To further improve the surface quality, the functions G642 (rounding function) and SOFT (jerk limitation) can be used. The commands must be written together at the beginning of the program.

Example:

```
Program code                          Comment
PROC ...
N10 COMPCAD SOFT G642                 ; Activating the COMPCAD compressor
N20 G01 X... Y... Z... F...           ; Traversing blocks 1 ... n
...
N1000 COMPOF                          ; Deactivation of the compressor
N1010 RET
```

Deactivation

All compressor functions are deactivated using the COMPOF command.

References

The programming of the compressor functions is described in:
Programming Manual Work Preparation

The use of the compressor function with active orientation transformation is described in:
Function Manual Special Functions; Multi-Axis Transformations (F2),
Section: Compression of the orientation

See also

Tolerance and compression of G0 blocks (Page 236)

3.5.2 Combine short spline blocks

Function

During the preparation of spline blocks, blocks with short lengths can always occur between blocks with long lengths. This can mean that the path velocity must always be significantly reduced before these short blocks.

With the "Combine short spline blocks" function, the spline blocks are prepared in such a way that blocks with short lengths are avoided and therefore traversing can be performed smoothly with a high path velocity.

Note

NC block compressor

The NC block compressor (COMPON, COMPCURV or COMPCAD) cannot be employed while compressing spline blocks, since with this only linear blocks can be compressed.

Availability

System	Availability
SINUMERIK 840D sl	Standard (basic scope)
SINUMERIK 828D	Option

Activation

The "Combine short spline blocks" function can be activated for the following spline types:

- BSPLINE
- BSPLINE/ORICURVE
- CSPLINE

The activation is done using machine data:

MD20488 $MC_SPLINE_MODE, bit <n> = <value> (setting for spline interpolation)

Bit	<value>	Meaning: "Combine short spline blocks" function ...
0	0	For BSPLINE not active
	1	For BSPLINE active
1	0	For BSPLINE/ORICURVE not active
	1	For BSPLINE/ORICURVE active
2	0	For CSPLINE not active
	1	For CSPLINE active

Supplementary conditions

- Spline blocks can only be combined if no other functions are programmed except traversing motions and feedrate. With, for example, auxiliary functions that are output on the PLC, the spline blocks cannot be combined.
- The maximum number of blocks that can be combined into a program section in succession, depends on the size of the memory available for blocks in the block preparation.

  MD28070 $MC_MM_NUM_BLOCKS_IN_PREP (number of blocks for block preparation)

Example

In order to attain a higher path velocity when executing the following program, the "Combine short spline blocks" function is activated for BSPLINE interpolation:

MD20488 $MC_SPLINE_MODE, Bit 0 = 1

```
Program code                                    Comment
PROC P1
N10 G1 G64 X0 Y0 Z0 F1000
N20 G91 F10000 BSPLINE
; BSPLINE interpolation with Combine short spline blocks from this point
N30 X0.001 Y0.001 Z0.001
N40 X0.001 Y0.001 Z0.001
N50 X0.001 Y0.001 Z0.001
N60 X0.001 Y0.001 Z0.001
N70 X0.001 Y0.001 Z0.001
N80 X0.001 Y0.001 Z0.001
...
N1000 M30
```

3.6 Contour/Orientation tolerance

Parameterization for the contour/orientation tolerance

The maximum permissible contour deviation (contour tolerance) and the maximum permissible angular deviation for the tool orientation (orientation tolerance) are defined in the machine data for every axis:

MD33100 $MA_COMPRESS_POS_TOL (maximum tolerance with compression)

The value set is valid both for the compressor functions and for the rounding functions with the exception of G641 (in this case, the distance to the block transition programmed with ADIS/ADISPOS applies).

Tolerance values from the following setting data may also be effective instead of MD33100:

SD42465 $SC_SMOOTH_CONTUR_TOL (maximum contour deviation)

SD42466 $SC_SMOOTH_ORI_TOL (maximum angular deviation of the tool orientation)

The manner in which the tolerance values from MD33100 and from the SD42645 and SD42466 setting data are to be taken into consideration is set as follows:

- For the rounding functions - via the decimal places in the machine data:

  MD20480 $MC_SMOOTHING_MODE (rounding behavior with G64x)

- For the compressor functions - via the ones position in the machine data:

  MD20482 $MC_COMPRESSOR_MODE (mode of compression)

Further machine and setting data which are of relevance in contour and orientation tolerance settings are:

- MD33120 $MA_PATH_TRANS_POS_TOL (max. deviation when rounding with G645)

  The value from MD33120 is effective when rounding block transitions with uniform tangents and non-uniform curvature (e.g. circle/straight line transition) with G645.

- SD42676 $SC_ORI_SMOOTH_TOL

  This setting data determines the tolerance when rounding the surface with OST.

- SD42678 $SC_ORISON_TOL

  This setting data determines the tolerance when smoothing the orientation with ORISON.

Programming the contour/orientation tolerance

The machine and setting data described here take effect when the program is started and determine the tolerances for all compressor functions, the rounding functions G642, G643, G645, OST, and the ORISON orientation smoothing.

However, the NC program can overwrite these configured tolerances. The NC programmer has the following commands at his/her disposal for this purpose:

Command	Syntax	Meaning
CTOL	CTOL=<value>	Contour tolerance
OTOL	OTOL=<value>	Orientation tolerance
ATOL[<axis>]	ATOL[<axis>]=<value>	Axis-specific tolerance

CTOL and OTOL have priority over ATOL.

Programming does not trigger a preprocessing stop. If possible, it does not interrupt NC block compression either.

The programmed values are valid until they are reprogrammed or deleted by being written with a negative value. They are also deleted at the end of a program, in the event of a channel reset, a mode group reset, an NCK reset (warm restart), and POWER ON (cold restart). On deletion of these values, the values from the machine and setting data are restored.

New values can be programmed and become effective in any block.

Note

The programmed tolerance also acts upon functions which are only implicitly dependent upon the tolerance. These are currently:

- Limiting the chord error in the setpoint value calculation
- The basic functions of the free-form surface mode

Note

The following rounding functions are not affected by the programming of CTOL, OTOL, and ATOL:

- Rounding orientation with OSD

  Reason: OSD does not use a tolerance, it uses a distance from the block transition.
- Rounding with G644

  Reason: G644 is not used for processing, it is used for optimizing tool changes and other movements in air.
- Rounding block transitions with uniform tangents and non-uniform curvature with G645

  G645 virtually always behaves like G642 and, thus, uses the programmed tolerances. The tolerance value from machine data MD33120 $MA_PATH_TRANS_POS_TOL is only used in uniformly tangential block transitions with a jump in curvature, e.g. a tangential circle/straight line transition. The rounding path at these points may also be located outside the programmed contour, where many applications are less tolerant. Furthermore, it generally takes a small, fixed tolerance to compensate for the sort of changes in curvature which need not concern the NC programmer.

Read tolerance values

For more advanced applications or for diagnostics, the currently valid tolerances for the compressor functions (COMPON, COMPCURV, COMPCAD), the smoothing types G642, G643, G645, OST, and the orientation smoothing ORISON can be read via system variables irrespective of how they might have come about.

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variables:

$AC_CTOL	Contour tolerance effective when the current main run record was preprocessed. If no contour tolerance is effective, $AC_CTOL will return the root from the sum of the squares of the tolerances of the geometry axes.
$AC_OTOL	Orientation tolerance effective when the current main run record was preprocessed. If no orientation tolerance is effective, $AC_OTOL will return the root from the sum of the squares of the tolerances of the orientation axes during active orientation transformation. Otherwise, it will return the value "-1".
$AA_ATOL[<axis>]	Axis tolerance effective when the current main run record was preprocessed. If no contour tolerance is active, $AA_ATOL[<geometry axis>] returns the contour tolerance divided by the root of the number of geometry axes. If an orientation tolerance and an orientation transformation are active $AA_ATOL[<orientation axis>] will return the orientation tolerance divided by the root of the number of orientation axes.

Note

If no tolerance values have been programmed, the $A variables will not be differentiated sufficiently to distinguish potential differences in the tolerances of the individual functions, since they can only declare one value.

Circumstances like this can occur if the machine data and the setting data set different tolerances for compressor functions, smoothing and orientation smoothing. The variables then return the greatest value prevailing with regard to the currently active functions.

If, for example, a compressor function is active with an orientation tolerance of 0.1° and ORISON orientation smoothing with 1°, the $AC_OTOL variable will return the value "1". If orientation smoothing is deactivated, only the value "0.1" will remain to be read.

- Without preprocessing stop in the part program via system variables:

$P_CTOL	Programmed contour tolerance
$P_OTOL	Programmed orientation tolerance
$PA_ATOL	Programmed axis tolerance

Note

If no tolerance values have been programmed, the $P variables return the value "-1".

3.7 Tolerance and compression of G0 blocks

Function

The function "Tolerance and compression of G0 blocks" allows rapid traverse motion to be executed faster.

It consists of the following components:

1. **Configuring/programming an independent tolerance factor for G0 motion**

   Using this factor, the tolerances for G0 motion can be set differently to the workpiece machining tolerances.

2. **Compressing G0 blocks**

   If this functionality is selected, for active NC block compression, in addition to traversing blocks with G1 (straight line interpolation), traversing blocks with G0 (rapid traverse) are compressed.

Effectiveness

1. The G0 tolerance factor is only effective, if:

- One of the following functions is active:
  - Compressor functions: COMPON, COMPCURV and COMPCAD
  - Smoothing functions: G642 and G645
  - Orientation smoothing: OST
  - Orientation smoothing: ORISON
  - Smoothing for path-relevant orientation: ORIPATH
- Several (≥ 2) consecutive G0 blocks in the part program.

  For a single G0 block, the G0 tolerance factor is not effective, as **at the transition** from a non G0 motion to a G0 motion (and vice versa), the **"lower tolerance"** always applies (workpiece machining tolerance)!

2. The compression of G0 blocks becomes effective:

- For NC block compression (COMPON, COMPCURV or COMPCAD).

Configuration

G0 tolerance factor

The G0 tolerance factor is set channel-specific with the machine data:

MD20560 $MC_G0_TOLERANCE_FACTOR (tolerance factor for G0)

The G0 tolerance factor can be both greater or less than 1.0. If the factor equals 1.0 (default value), then the same tolerances for G0 blocks are effective as for non-G0 blocks. Normally, the G0 tolerance factor is set to ≥ 1.0.

Compressing G0 blocks

The compression of G0 blocks is set for specific channels using the **hundreds position** in the machine data:

MD20482 $MC_COMPRESSOR_MODE (mode of compression)

Value	Meaning
0xx	Circular blocks and G0 blocks are **not** compressed.
1xx	Circular blocks are compressed. Only COMPCAD.
2xx	G0 blocks are compressed. See also MD20560 $MC_G0_TOLERANCE_FACTOR or NC command `STOLF`
3xx	Circular blocks and G0 blocks are compressed.

A more detailed description of MD20482 can be found in:
References:
Function Manual Special Functions; Multi-Axis Transformations (F2),
Section: Compression of the orientation

Programming

The tolerance factor set using MD20560 $MC_G0_TOLERANCE_FACTOR can be temporarily overwritten by programming `STOLF` in the part program:

Syntax: `STOLF=<…>`

Example:

```
Program code                    Comment
COMPCAD G645 G1 F10000          ; Compressor function COMPCAD
X... Y... Z...                  ; The machine and setting data apply here.
X... Y... Z...
X... Y... Z...
G0 X... Y... Z...
G0 X... Y... Z...               ; Machine data $MC_G0_TOLERANCE_FACTOR (e.g. =3), is
                                  effective here, i.e. a smoothing tolerance of
                                  $MC_G0_TOLERANCE_FACTOR*$MA_COMPRESS_POS_TOL.
CTOL=0.02
STOLF=4
G1 X... Y... Z...               ; A contour tolerance of 0.02 mm is applied starting
                                  from here.
X... Y... Z...
X... Y... Z...
G0 X... Y... Z...
X... Y... Z...                  ; From here, a G0 tolerance factor of 4 applies, i.e. a
                                  contour tolerance of 0.08 mm.
```

The value in MD20560 is not changed by programming the tolerance factor. After a reset or end of part program, the value set using MD20560 is effective again.

Reading the tolerance factor

The G0 tolerance factor, effective in the part program or in the actual IPO block, can be read using system variables.

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variable:

$AC_STOLF	Active G0 tolerance factor G0 tolerance factor, which was effective when processing the actual main run block.

- Without preprocessing stop in the part program via the system variable:

$P_STOLF	Programmed G0 tolerance factor

If no value with `STOLF` is programmed in the active part program, then these two system variables supply the value set using MD20560 $MC_G0_TOLERANCE_FACTOR.

If no rapid traverse (G0) is active in a block, then these system variables always supply a value of 1.

3.8 RESET behavior

MD20150

The channel-specific initial state is activated via a RESET for G function groups:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

The following G function groups are of relevance to "continuous-path mode, exact stop, LookAhead":

- Group 10: Exact stop - continuous-path mode
- Group 12: Block-change criterion for exact stop
- Group 21: Acceleration profile
- Group 30: NC block compression
- Group 59: Dynamic response mode for path interpolation

For detailed information on setting initial states, see Section "K1: Mode group, channel, program operation, reset response (Page 489)".

3.9 Supplementary conditions

3.9.1 Block change and positioning axes

If path axes are traversed in continuous path mode in a part program, traversing positioning axes can also simultaneously affect both the response of the path axes and the block change.

A detailed description of the positioning axes can be found in:
References:
Function Manual, Extended Functions; Positioning axes (P2)

3.9.2 Block change delay

Even if all path axes and special axes traversing in the part program block have satisfied their specific block transition criteria, the block change can still be delayed due to other unsatisfied conditions and/or active functions:

Examples:

- Missing auxiliary function acknowledgement by the PLC
- Non-existent following blocks
- Active function "Empty buffer"

Effects

If a block change cannot be executed in continuous path mode, all axes programmed in this part program block (except cross-block traversing special axes) are stopped. In this case, contour errors do not occur.

The stopping of path axes **during machining** can cause undercuts on the workpiece surface.

3.9.3 Rounding and repositioning (REPOS)

Repositioning within the rounding area

If the traversing motion of the path axes within the rounding area is interrupted for traversing blocks with programmed rounding (part program command G641, G642, G643, G644 or G645), repositioning occurs as follows in the event of a subsequent REPOS operation, depending on the current REPOS mode:

REPOS mode	REPOS end point
RMBBL	Block start of interrupted traversing block
RMIBL	Block end of interrupted traversing block
RMEBL	Block end of interrupted traversing block
RMNBL	Block end of interrupted traversing block

Example

Two traversing blocks N10 and N20 with programmed rounding G641. In the rounding area, the traversing motion is interrupted and the axes are subsequently traversed, e.g. manually to the REPOS starting point. Repositioning on the contour takes place differently, depending on the active REPOS mode.

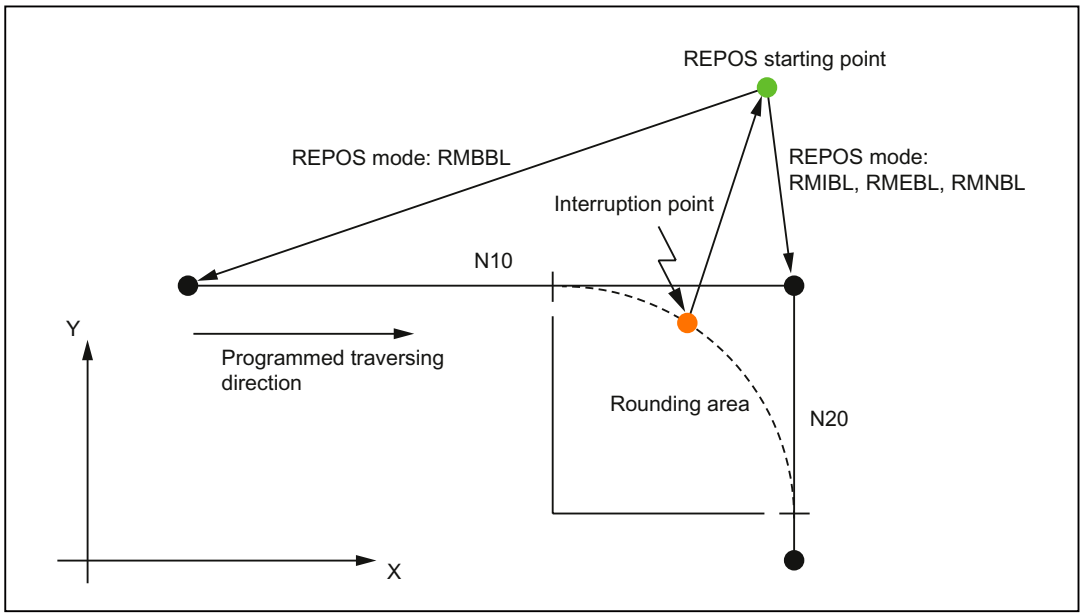


Figure 3-13 Example of rounding and repositioning

3.10 Data lists

3.10.1 Machine data

3.10.1.1 General machine data

Number	Identifier: $MN_	Description
10110	PLC_CYCLE_TIME_AVERAGE	Average PLC acknowledgment time
10680	MIN_CONTOUR_SAMPLING_TIME	Minimum contour sampling time
10682	CONTOUR_SAMPLING_FACTOR	Contour sampling factor
10712	NC_USER_CODE_CONF_NAME_TAB	List of reconfigured NC commands
12030	OVR_FACTOR_FEEDRATE	Evaluation of the path feed override switch
12100	OVR_FACTOR_LIMIT_BIN	Limit for binarycoded override switch
18360	MM_EXT_PROG_BUFFER_SIZE	FIFO buffer size for execution from external source (DRAM)

3.10.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Initial setting of the groups
20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of NC block for compression
20172	COMPRESS_VELO_TOL	Maximum permissible deviation from path feed for compression
20400	LOOKAH_USE_VELO_NEXT_BLOCK	LookAhead following block velocity
20430	LOOKAH_NUM_OVR_POINTS	Number of override switch points for LookAhead
20440	LOOKAH_OVR_POINTS	Override switch points for LookAhead
20443	LOOKAH_FFORM	Activating the extended LookAhead
20450	LOOKAH_RELIEVE_BLOCK_CYCLE	Relief factor for the block cycle time
20460	LOOKAH_SMOOTH_FACTOR	Smoothing factor for LookAhead
20462	LOOKAH_SMOOTH_WITH_FEED	Smoothing with programmed feed
20465	ADAPT_PATH_DYNAMIC	Adaptation of path dynamic response
20480	SMOOTHING_MODE	Rounding behavior with G64x
20482	COMPRESSOR_MODE	Compressor mode
20488	SPLINE_MODE	Setting for spline interpolation
20490	IGNORE_OVL_FACTOR_FOR_ADIS	G641/G642 independent of the overload factor
20550	EXACT_POS_MODE	Exact-stop conditions with G0/G1
20560	G0_TOLERANCE_FACTOR	G0 tolerance factor
20600	MAX_PATH_JERK	Pathrelated maximum jerk

Number	Identifier: $MC_	Description
20602	CURV_EFFECT_ON_PATH_ACCEL	Influence of path curvature on path dynamic response
20603	CURV_EFFECT_ON_PATH_JERK	Influence of path curvature on path jerk
20606	PREPDYN_SMOOTHING_ON	Activation of the curvature smoothing
28060	MM_IPO_BUFFER_SIZE	Number of NC blocks in IPO buffer (DRAM)
28070	MM_NUM_BLOCKS_IN_PREP	Number of NC blocks for block preparation (DRAM)
28520	MM_MAX_AXISPOLY_PER_BLOCK	Maximum number of axis polynomials per block
28530	MM_PATH_VELO_SEGMENTS	Number of storage elements for limiting path velocity in block
28533	MM_LOOKAH_FFORM_UNITS	Storage for the extended LookAhead
28540	MM_ARCLENGTH_SEGMENTS	Number of storage elements for arc length function representation per block
28610	MM_PREPDYN_BLOCKS	Number of blocks for velocity preparation

3.10.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32310	MAX_ACCEL_OVL_FACTOR	Overload factor for axial velocity jumps
32431	MAX_AX_JERK	Maximum axial jerk when traversing along the path
32432	PATH_TRANS_JERK_LIM	Maximum axial jerk at the block transition in continuous-path mode
32433	SOFT_ACCEL_FACTOR	Scaling of acceleration limitation for SOFT
32434	G00_ACCEL_FACTOR	Scaling of acceleration limitation for G00
32435	G00_JERK_FACTOR	Scaling of axial jerk limitation for G00
32440	LOOKAH_FREQUENCY	Smoothing limit frequency for LookAhead
33100	COMPRESS_POS_TOL	Maximum deviation with compensation
33120	PATH_TRANS_POS_TOL	Maximum deviation when rounding with G645
35240	ACCEL_TYPE_DRIVE	DRIVE acceleration characteristic for axes on/off
36000	STOP_LIMIT_COARSE	Exact stop coarse
36010	STOP_LIMIT_FINE	Exact stop fine
36012	STOP_LIMIT_FACTOR	Exact stop coarse/fine factor and zero speed monitoring
36020	POSITIONING_TIME	Delay time exact stop fine

3.10.2 Setting data

3.10.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42465	SMOOTH_CONTUR_TOL	Max. contour deviation during rounding
42466	SMOOTH_ORI_TOL	Max. deviation of the tool orientation during rounding
42470	CRIT_SPLINE_ANGLE	Core limit angle, compressor
42475	COMPRESS_CONTUR_TOL	Maximum contour deviation in the compressor

3.10.3 Signals

3.10.3.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
All axes stationary	DB21, ...DBX36.3	DB3300.DBX4.3

3.10.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Position reached with exact stop coarse	DB31, ...DBX60.6	DB390x.DBX0.6
Position reached with exact stop fine	DB31, … .DBX60.7	DB390x.DBX0.7

B2: Acceleration

4

4.1 Brief description

4.1.1 General

Scope of functions

The Description of Functions covers the following sub-functions:

- Acceleration
- Jerk
- Kneeshaped acceleration characteristic

Acceleration and jerk

The effective acceleration and jerk can be optimally matched to the machine and machining situation concerned using axis- and channel-specific programmable maximum values, programmable acceleration profiles in part programs and synchronized actions, and dynamic adaptations and limitations.

Kneeshaped acceleration characteristic

The knee-shaped acceleration characteristic means that, in the case of machine axes featuring a motor (in particular stepper motors) with a torque characteristic that is highly dependent upon speed, acceleration can be set at the level required to ensure optimum utilization of the motor whilst at the same time protecting it against overload.

4.1.2 Features

Acceleration

Axis-specific functions:

- Programmable maximum acceleration value
- Acceleration profile that can be selected via part-program instruction:

  Acceleration without jerk limitation (BRISKA)
- Setting of maximum value using part-program instruction (ACC)
- Specific maximum value for programmed rapid traverse (G00).
- Specific maximum value for traverse with active jerk limitation
- Excessive acceleration for non-tangential block transitions

Channel-specific functions:

- Acceleration profile that can be selected via part-program instruction:

  Acceleration without jerk limitation (BRISK)
- Programmable constant travel time for the purpose of avoiding extreme sudden acceleration
- Programmable acceleration margin for overlaid traversing
- Adjustable acceleration limitation
- Adjustable acceleration for specific real-time events
- Programmable acceleration margin for radial acceleration

Jerk

Axis-specific functions:

- Acceleration profile that can be selected via part-program instruction:

  Acceleration with jerk limitation (SOFTA)
- Programmable maximum jerk value for single-axis interpolation
- Programmable maximum jerk value for path interpolation

Channel-specific functions:

- Acceleration profile that can be selected via part-program instruction:

  Acceleration with jerk limitation (SOFT)
- Adjustable jerk limitation
- Adjustable path jerk for specific real-time events
- Specific maximum value for programmed rapid traverse (G00)
- Excessive jerk for block transitions without constant curvature

Kneeshaped acceleration characteristic

A knee-shaped acceleration characteristic is parameterized using the following characteristic data:

- Maximum velocity v_{max}
- Maximum acceleration a_{max}
- Creep velocity v_{red}
- Creep acceleration a_{red}
- Nature of the acceleration reduction (constant, hyperbolic, linear)

4.2 Functions

4.2.1 Acceleration without jerk limitation (BRISK/BRISKA) (channel/axis-specific)

4.2.1.1 General Information

General Information

In the case of acceleration without jerk limitation (jerk = infinite) the maximum value is applied for acceleration immediately. As regard to acceleration with jerk limitation, it differs in the following respects:

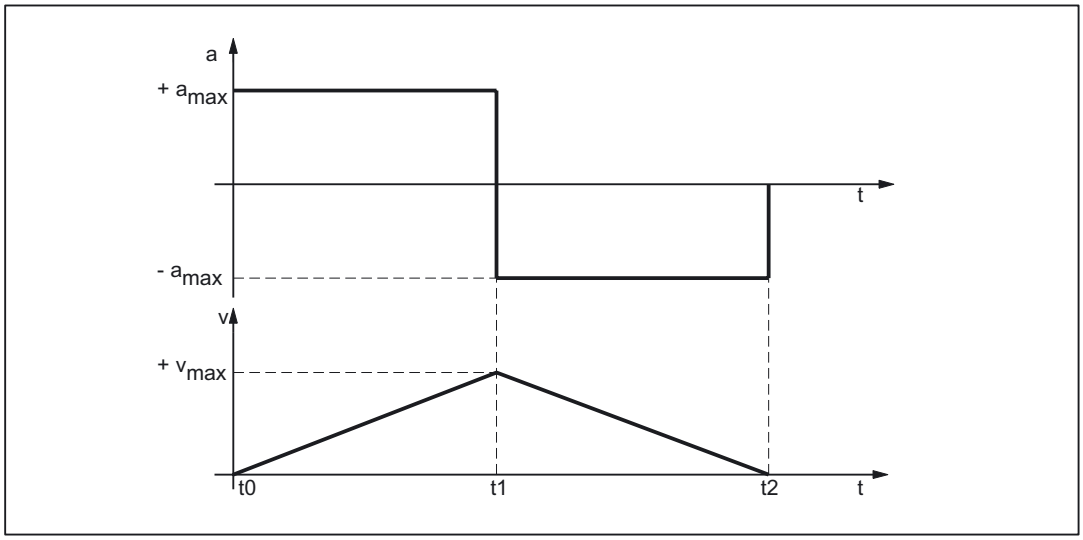
- Advantages

  Shorter processing times with the same maximum values for velocity and acceleration.

- Disadvantages

  Increased load on the machine's mechanical components and risk of inducing high-frequency and difficult-to-control mechanical vibrations.

Acceleration profile



a_{max}: Maximum acceleration value
v_{max}: Maximum velocity value
t: Time

Figure 4-1 Velocity and acceleration schematic for stepped acceleration profile

The following features of the acceleration profile can be identified from the figure above:

- Time: t_0

  Sudden acceleration from 0 to $+a_{max}$
- Interval: t_0 - t_1

  Constant acceleration with $+a_{max}$; linear increase in velocity
- Time: t_1

  Sudden acceleration from 2 * a_{max} with immediate switchover from acceleration to braking

  Note

  The sudden acceleration can normally be avoided by specifying a constant velocity time (see Section "Constant travel time (channel-specific) (Page 251)").
- Interval: t_1 - t_2

  Constant acceleration with $-a_{max}$; linear decrease in velocity

4.2.1.2 Parameterization

Maximum axial acceleration for path motions

The maximum axial acceleration for path motions can be set for the specific technology for each machine axis via the following machine data:

MD32300 $MA_MAX_AX_ACCEL[<parameter set index>]

With <parameter set index> = 0, 1, 2 ... (max. parameter set number - 1)

For the technology-specific parameter sets, see Section "Dynamic response mode for path interpolation (Page 219)".

The path parameters are calculated by the path planning of the preprocessing so that the parameterized maximum values of the machine axes involved in the path are not exceeded.

Note

It is possible for the maximum value to be exceeded in connection with specific machining situations (see Section "Acceleration matching (ACC) (axis-specific) (Page 253)" and "Path acceleration for real-time events (channel-specific) (Page 256)").

Maximum axial acceleration for positioning axis motions

With positioning axis motions, one of the two following maximum values is effective depending on the set positioning axis dynamic response mode:

- MD32300 $MA_MAX_AX_ACCEL [0] (maximum axial acceleration for path motions in the dynamic response mode DYNNORM)
- MD32300 $MA_MAX_AX_ACCEL [1] (maximum axial acceleration for path motions in the dynamic response mode DYNPOS)

The positioning axis dynamic response mode is set in the NC-specific machine data:

MD18960 $MN_POS_DYN_MODE = <mode>

<mode>	Meaning
0	Effective maximum axial acceleration: MD32300 $MA_MAX_AX_ACCEL[0]
1	Effective maximum axial acceleration: MD32300 $MA_MAX_AX_ACCEL[1]

Maximum axial acceleration for JOG motions

For JOG mode, a JOG-specific maximum acceleration value can be configured for each machine axis (see Section "Acceleration and jerk for JOG motions (Page 288)").

4.2.1.3 Programming

Path acceleration without jerk limitation (BRISK)

Syntax

```
BRISK
```

Functionality

The BRISK part-program instruction is used to select the "without jerk limitation" acceleration profile for the purpose of path acceleration.

G group: 21

Effective: Modal

Reset response

The channel-specific initial setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

Supplementary conditions

If the acceleration profile is changed in a part program during machining (BRISK/SOFT) an exact stop is performed at the end of the block.

Single-axis acceleration without jerk limitation (BRISKA)

Syntax

BRISKA (*axis{,axis}*)

Function

The BRISKA part-program instruction is used to select the "without jerk limitation" acceleration profile for single-axis movements (JOG, JOG/INC, positioning axis, reciprocating axis, etc.).

G group: -

Effective: Modal

Axis:

- Value range: Axis identifier for channel axes

Axis-specific initial setting

Acceleration without jerk limitation can be set as the axis-specific initial setting for single-axis movements:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE = FALSE

Reset response

The axis-specific initial setting is activated via a reset:

MD32420 $MA_JOG_AND_POS_ENABLE

4.2.2 Constant travel time (channel-specific)

4.2.2.1 General Information

Overview

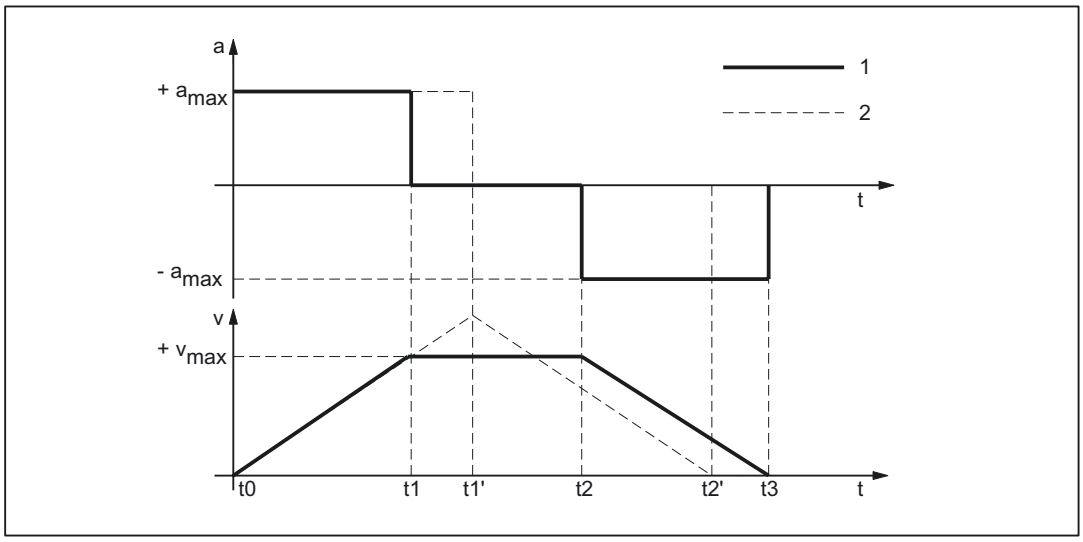
In the case of acceleration without jerk limitation, sudden acceleration of 2 * a_{max} occurs on switchover between acceleration and braking. In order to avoid this sudden acceleration, a channel-specific constant travel time can be programmed. The constant travel time defines the time taken to traverse between the acceleration and braking phases at constant velocity:

MD20500 $MC_CONST_VELO_MIN_TIME (minimum time with constant velocity)

Note

The constant travel time is ineffective:

- Active function: Look Ahead
- In traversing blocks with a travel time that is less or equal to the interpolation cycle time.



1: Characteristic with constant travel time
2: Characteristic without constant travel time
a_{max}: Maximum acceleration value
v_{max}: Maximum velocity value
t: Time

Figure 4-2 Schematic for abrupt acceleration

The effect of the constant travel time can be seen from the figure above:

- Time: t_1

  End of acceleration phase with sudden acceleration 1 * a_{max}

- Interval: t_1 - t_2

  Acceleration 0; constant velocity using the parameterized constant travel time

- Time: t_2

  Start of braking phase with sudden acceleration 1 * a_{max}

The times t_0, t_1' and t_2' indicate the characteristic that would have been produced had no constant travel time been defined.

4.2.2.2 Parameterization

The constant travel time is parameterized for specific channels using machine data:

MD20500 $MC_CONST_VELO_MIN_TIME
(minimum time with constant velocity)

4.2.3 Acceleration matching (ACC) (axis-specific)

4.2.3.1 General Information

Function

A part-program instruction (ACC) can be used to match the acceleration of specific axes to the current machining situation. The range used for this purpose is anywhere between greater than 0% and less than or equal to 200% of the maximum value programmed in the machine data.

Effective

Effective	Acceleration matching is effective for all types of interpolation in AUTOMATIC and MDA operating modes as well as with dry-run feed.
Ineffective	Acceleration matching is ineffective in JOG and JOG/REF (reference point approach) operating modes. Acceleration matching is also ineffective if the machine axes have been brought to a standstill via a quick stop due to the detection of a fault (setpoint = 0).

4.2.3.2 Programming

Syntax

`ACC[axis]` = *adjustment factor*

Functionality

The ACC part-program instruction is used to adjust the maximum acceleration value of a machine axis.

Axis:

- Value range: Axis identifier for the channel's machine axes

Adjustment factor:

- Value range: 0 < adjustment factor ≤ 200
- Unit: Per cent

Deactivate: `ACC[`*axis*`] = 100`

Effective: Modal

Reset response

The behavior during channel RESET or M30 can be controlled via MD32320 $MA_DYN_LIMIT_RESET_MASK :

Bit 0: 0 The programmed ACC value is reduced to 100 % with channel RESET/M30 .

Bit 0: 1 The programmed ACC value is retained beyond channel RESET/M30.

4.2.4 Acceleration margin (channel-specific)

4.2.4.1 General Information

General information

Under normal circumstances, preprocessing makes maximum use of the parameterized maximum values of the machine axes for the purpose of path acceleration. In order that an acceleration margin may be set aside for overlaid movements, e.g., within the context of the "Rapid lift away from the contour" function, path acceleration can be reduced by a programmable factor. When, for example, a factor of 0.2 is applied, preprocessing will only use 80% of the maximum possible path acceleration. 20% is set aside as an acceleration margin for overlaid movements.

4.2.4.2 Parameterization

Parameters for the acceleration margin are assigned for each channel by means of machine datum:
MD20610 $MC_ADD_MOVE_ACCEL_RESERVE
(acceleration margin for overlaid motions)

4.2.5 Path-acceleration limitation (channel-specific)

4.2.5.1 General Information

General Information

To enable a flexible response to the machining situations concerned, setting data can be used to limit the path acceleration calculated during preprocessing for specific channels:

SD42500 $SC_SD_MAX_PATH_ACCEL (maximum path acceleration)

The value specified in the setting data is only taken into account if it is smaller than the path acceleration calculated during preprocessing.

The limitation must be activated for specific channels using setting data:

SD42502 $SC_IS_SD_MAX_PATH_ACCEL = TRUE

4.2.5.2 Parameterization

Parameterization is carried out for specific channels using setting data:

SD42500 $SC_SD_MAX_PATH_ACCEL (maximum path acceleration)

SD42502 $SC_IS_SD_MAX_PATH_ACCEL (activation of path-acceleration limitation)

4.2.5.3 Programming

Limit value

Syntax

`$SC_SD_MAX_PATH_ACCEL =` *limit value*

Functionality

The path-acceleration limitation can be adjusted for the situation by programming the setting data.

Limit value:

- Value range: ≥ 0
- Unit: m/s^2

Application:

- Part program
- Static synchronized action

Switch ON/OFF

Syntax

```
$SC_IS_SD_MAX_PATH_ACCEL = value
```

Functionality

The path-acceleration limitation can be activated/deactivated by programming the setting data.

Parameter: *Value*

- Value range: TRUE, FALSE

Application:

- Part program
- Static synchronized action

4.2.6 Path acceleration for real-time events (channel-specific)

4.2.6.1 General Information

General Information

So that no compromise has to be made between machining-optimized acceleration on the one hand and time-optimized acceleration in connection with the following real-time events on the other:

- NC Stop / NC Start
- Changing the feedrate override
- Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function

For the real-time events mentioned above, the path acceleration can be specified using a channel-specific system variable:

```
$AC_PATHACC = path acceleration
```

Real-time event acceleration will only be active for the duration of the change in velocity in respect of one of the real-time events specified above.

Limitation

If the specified path acceleration exceeds the capabilities of the machine axes that are of relevance for the path, a limit will be imposed on the path acceleration within the controller so that the resulting axial acceleration (a_{res}) is restricted to less than 2x the parameterized maximum axial value (a_{max}).

$a_{res} = 2 * a_{max,}$ with a_{max} = MD32300 $MA_MAX_AX_ACCEL

Note

Path acceleration for real-time events is enabled, irrespective of the radial acceleration.

Effectiveness

Effective	Real-time event acceleration is only enabled in AUTOMATIC and MDA operating modes in conjunction with the following real-time events: • NC Stop / NC Start • Override changes • Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function
Not effective	Path acceleration for real-time events is ineffective for changes in path velocity that are attributable to path planning during preprocessing for the channel, such as contour curvatures, corners, kinematic transformation limitations, etc. Real-time-event path acceleration is ineffective if the programmed value is smaller than the path acceleration calculated during preprocessing for the path section concerned.

Programming

For information about programming system variables in the part program or synchronized actions, see Section "Programming (Page 258)".

4.2.6.2 Programming

Syntax

$AC_PATHACC = *path acceleration*

Functionality

Real-time-event path acceleration is set via the channel-specific system variables.

Parameter: *Path acceleration*

- Value range: Path acceleration ≥ 0
- Unit: m/s²

Deactivation: $AC_PATHACC = 0

Application:

- Part program
- Static synchronized action

Reset response

Real-time-event path acceleration is deactivated on reset.

Supplementary conditions

Programming $AC_PATHACC in the part program automatically triggers a preprocessing stop with REORG (STOPRE).

4.2.7 Acceleration with programmed rapid traverse (G00) (axis-specific)

4.2.7.1 General Information

Frequently, the acceleration for the machine axes involved in the machining process must be set lower than the machine's performance capability officially allows because of the supplementary conditions associated with the specific process concerned.

For time-optimized traversing of the machine axes with programmed rapid traverse (part-program instruction G00), a specific maximum value can be programmed for the axis-specific acceleration.

JOG setup mode

This function does not affect acceleration in respect of a rapid traverse override in JOG setup mode.

4.2.7.2 Parameterization

The maximum value for axis-specific acceleration with programmed rapid traverse is parameterized (G00) using the axis-specific machine data:

MD32434 $MA_G00_ACCEL_FACTOR
(scaling of the acceleration limitation with G00)

This is used to generate the maximum value for axis-specific acceleration with programmed rapid traverse (G00) that is taken into account by the path planning component during preprocessing:

Acceleration[axis] =
MD32300 $MA_MAX_AX_ACCEL * MD32434 $MA_G00_ACCEL_FACTOR

4.2.8 Acceleration with active jerk limitation (SOFT/SOFTA) (axis-specific)

4.2.8.1 General Information

Function

Compared with acceleration without jerk limitation, acceleration with jerk limitation results in a certain degree of time loss, even when the same maximum acceleration value is used. To compensate for this time loss, a specific maximum value can be programmed for the axis-specific acceleration as far as traversing of the machine axes with active jerk limitation (SOFT/SOFTA) is concerned.

The maximum value for acceleration with active jerk limitation is parameterized using a factor calculated in relation to the axis-specific maximum value. This is used to generate the maximum value for axis-specific acceleration with active jerk limitation that is taken into account by the path planning component during preprocessing:

Acceleration[axis] =
MD32300 $MA_MAX_AX_ACCEL * MD32433 $MA_SOFT_ACCEL_FACTOR

4.2.8.2 Parameterization

The maximum value for acceleration with active jerk limitation (SOFT/SOFTA) is parameterized using the axis-specific machine data:

MD32434 $MA_SOFT_ACCEL_FACTOR
(scaling of the acceleration limitation with SOFT)

4.2.9 Excessive acceleration for non-tangential block transitions (axis-specific)

4.2.9.1 General Information

Function

In the case of non-tangential block transitions (corners), the programmable controller may have to decelerate the geometry axes significantly in order to ensure compliance with the parameterized axis dynamics. For the purpose of reducing/avoiding deceleration in connection with non-tangential block transitions, a higher level of axis-specific acceleration can be enabled.

Excessive acceleration is parameterized using a factor calculated in relation to the axis-specific maximum value. This is used to generate the maximum value for axis-specific acceleration with non-tangential block transitions that is taken into account by the path planning component during preprocessing:

Acceleration[axis] =
MD32300 $MA_MAX_AX_ACCEL * MD32310 $MA_MAX_ACCEL_OVL_FACTOR

4.2.9.2 Parameterization

Excessive acceleration for non-tangential block transitions is parameterized using the axis-specific machine data:
MD32310 $MA_MAX_ACCEL_OVL_FACTOR
(overload factor for velocity jumps)

4.2.10 Acceleration margin for radial acceleration (channel-specific)

4.2.10.1 General Information

Overview

In addition to the path acceleration (tangential acceleration), radial acceleration also has an effect on curved contours. If this is not taken into account during parameterization of the path parameters, the effective axial acceleration during acceleration and deceleration on the curved contour can, for a short time, reach 2x the maximum value.

Effective axial acceleration =
Path acceleration + radial acceleration =
2 * (MD32300 $MA_MAX_AX_ACCEL)

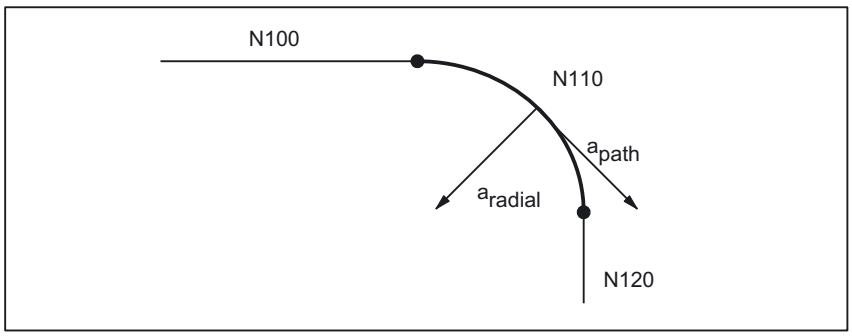


Figure 4-3 Radial and path acceleration on curved contours

The channel-specific machine data:
MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL
(influence of path curvature on dynamic path response)
can be used to set the proportion of the axis-specific acceleration that is to be taken into account for radial acceleration.

When, for example, a value of 0.75 is applied, 75% of the axis-specific acceleration will be made available for radial acceleration and 25% for path acceleration.

The corresponding maximum values are generally calculated as follows:

Radial acceleration =
MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL * MD32300 $MA_MAX_AX_ACCEL

Path acceleration =
(1 - MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL) * MD32300 $MA_MAX_AX_ACCEL

Example

The following machine parameters apply:

- MD32300 $MA_MAX_AX_ACCEL for all geometry axes: 3 m/s
- Maximum path velocity with a path radius of 10 mm due to mechanical constraints of the machine: 5 m/min.

The radial acceleration is calculated as follows:

$$a_{radial} = \frac{v^2_{path}\ [m/min]}{r\ [mm] * 3.6\ [m/s^2]} = \frac{5^2}{10 * 3.6} = 0.694\ m/s^2$$

The acceleration margin is set as follows:

$$\text{MD20602 \$MC\_CURV\_EFFECT\_ON\_PATH\_ACCEL} = \frac{a_{radial}\ [m/s^2]}{\text{MD32300 \$MA\_MAX\_AX\_ACCEL}\ [m/s^2]} = \frac{0.694}{3} \approx 0.23$$

Linear motions

The acceleration margin referred to above is ineffective in the case of linear motions (linear interpolation) without active kinematic transformation.

4.2.10.2 Parameterization

The proportion of maximum available axis acceleration to be taken into account as an acceleration margin for radial acceleration on curved contours is parameterized using the channel-specific machine data:

MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL
(influence of path curvature on dynamic path response)

4.2.11 Jerk limitation with path interpolation (SOFT) (channel-specific)

4.2.11.1 General Information

Overview

As far as the functionality described in the rest of this document is concerned, constant acceleration, i.e., acceleration with jerk limitation (jerk = infinite value), is the assumed acceleration profile. In the case of acceleration with jerk limitation, linear interpolation is applied in respect of acceleration from 0 to the maximum value.

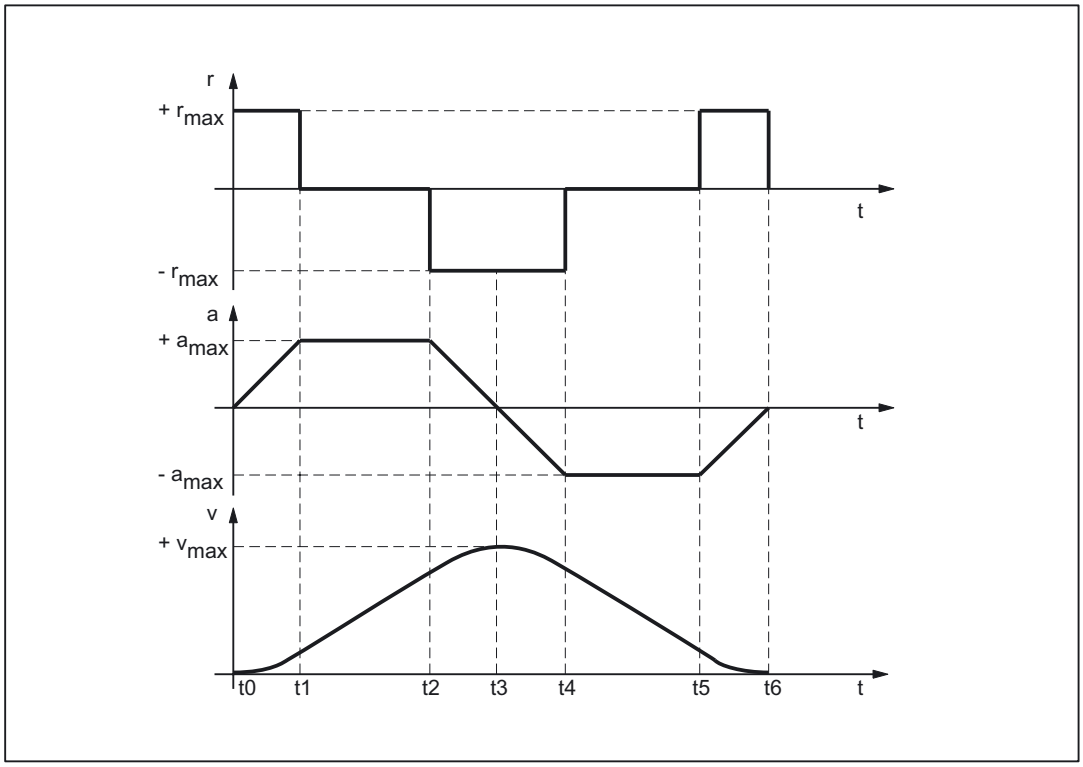
Advantages

Minimal load on the machine's mechanical components and low risk of high-frequency and difficult-to-control mechanical vibrations thanks to constant excessive acceleration.

Disadvantages

Longer machining times compared with stepped acceleration profile when the same maximum velocity and acceleration values are used.

Acceleration profile



r_{max}: Maximum jerk value
a_{max}: Maximum acceleration value
v_{max}: Maximum velocity value
t: Time

Figure 4-4 Jerk, acceleration and velocity schematic with jerk limitation acceleration profile

The following features of the acceleration profile can be identified from the figure above:

- Interval: t_0 - t_1

  Constant jerk with $+r_{max}$; linear increase in acceleration; quadratic increase in velocity
- Interval: t_1 - t_2

  Constant acceleration with $+a_{max}$; linear increase in velocity
- Interval: t_2 - t_3

  Constant jerk with $-r_{max}$; linear decrease in acceleration; quadratic decrease in excessive velocity until maximum value $+v_{max}$ is reached

- Interval: t_3 - t_4

  Constant jerk with $+r_{max}$; linear increase in braking acceleration; quadratic decrease in velocity

- Interval: t_4 - t_5

  Constant braking acceleration with $-a_{max}$; linear decrease in velocity

- Interval: t_5 - t_6

  Constant jerk with $-r_{max}$; linear decrease in braking acceleration; quadratic decrease in velocity reduction until zero velocity is reached v = 0

4.2.11.2 Parameterization

Maximum jerk value for path motions (axis-specific)

The maximum axial jerk for path motions can be set for the specific technology for each machine axis via the following machine data:

MD32431 $MA_MAX_AX_JERK[<parameter set index>]

With <parameter set index> = 0, 1, 2 ... (max. parameter set number - 1)

For the technology-specific parameter sets, see Section "Dynamic response mode for path interpolation (Page 219)".

The path parameters are calculated by the path planning of the preprocessing so that the parameterized maximum values of the machine axes involved in the path are not exceeded.

Note

It is possible for the maximum value to be exceeded in connection with specific machining situations (see Section "Path jerk for real-time events (channel-specific) (Page 270)").

Maximum jerk value for path motions (channel-specific)

In addition to the axis-specific setting, the maximum jerk value can also be specified as channel-specific path parameter via the following machine data:

MD20600 $MC_MAX_PATH_JERK (path-related maximum jerk)

In order to exclude the mutual influencing of axis and channel-specific maximum jerk values, the channel-specific maximum value must be set to a value greater than the axial maximum values.

4.2.11.3 Programming

Syntax

```
SOFT
```

Functionality

The SOFT part-program instruction is used to select the acceleration profile with jerk limitation for the traversing operations of geometry axes in the channel.

G group: 21

Effective: Modal

Reset response

The channel-specific initial setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

Boundary conditions

If the acceleration mode is changed in a part program during machining (BRISK ↔ SOFT), a block change is performed at the point of transition with an exact stop at the end of the block, even in continuous-path mode.

4.2.12 Jerk limitation with single-axis interpolation (SOFTA) (axis-specific)

4.2.12.1 Parameterization

Initial setting for axial jerk limitation

Acceleration with jerk limitation can be set as the axial initial setting:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE== TRUE

Maximum axial jerk for positioning axis motions

When traversing positioning axes with active jerk limitation, the value from one of the following machine data takes effect as maximum axial jerk:

- MD32430 $MA_JOG_AND_POS_MAX_JERK (maximum axial jerk for positioning axis motions)
- MD32431 $MA_MAX_AX_JERK [0] (maximum axial jerk for path motions in the dynamic response mode DYNNORM)
- MD32431 $MA_MAX_AX_JERK [1] (maximum axial jerk for path motions in the dynamic response mode DYNPOS)

The machine data to be used is determined by the set positioning axis dynamic response mode:

MD18960 $MN_POS_DYN_MODE = <mode>

<mode>	Meaning
	The following is effective as maximum axial jerk for positioning axis motions:
0	MD32430 $MA_JOG_AND_POS_MAX_JERK With active G75/G751 (fixed-point approach): MD32431 $MA_MAX_AX_JERK[**0**]
1	MD32431 $MA_MAX_AX_JERK[**1**]

Maximum axial jerk for JOG motions

For JOG mode, a JOG-specific maximum jerk value can be configured for each machine axis (see Section "Acceleration and jerk for JOG motions (Page 288)").

4.2.12.2 Programming

Syntax

SOFTA (*Axis* {*Axis*})

Functionality

The SOFTA part-program instruction is used to select acceleration with jerk limitation for single-axis movements (positioning axis, reciprocating axis, etc.)

G group: -

Effective: modal

Axis:

- Value range: Axis identifier for channel axes

Axis-specific initial setting

Acceleration with jerk limitation can be set as the axis-specific initial setting for single-axis movements:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE = TRUE

Reset response

The axis-specific initial setting is activated via a reset:

MD32420 $MA_JOG_AND_POS_ENABLE

4.2.13 Path-jerk limitation (channel-specific)

4.2.13.1 General Information

Overview

To enable a flexible response to the machining situations concerned, setting data can be used to limit the path jerk calculated during preprocessing for specific channels:

SD42510 $SC_SD_MAX_PATH_JERK (maximum path jerk)

The value specified in the setting data is only taken into account in the channel if it is smaller than the path jerk calculated during preprocessing.

The limitation must be activated for specific channels using setting data:

SD42512 $SC_IS_SD_MAX_PATH_JERK = TRUE

4.2.13.2 Parameterization

Parameterization is carried out for specific channels using setting data:

SD42510 $SC_SD_MAX_PATH_JERK (maximum path jerk)

SD42512 $SC_IS_SD_MAX_PATH_JERK
(activation of path-jerk limitation)

4.2.13.3 Programming

Maximum path jerk

Syntax

$SC_SD_MAX_PATH_JERK = *jerk value*

Functionality

The path-jerk limitation can be adjusted for the situation by programming the setting data.

Jerk value:

- Value range: ≥ 0
- Unit: m/s^3

Application:

- Part program
- Static synchronized action

Switch ON/OFF

Syntax

$SC_IS_SD_MAX_PATH_JERK = *value*

Functionality

The path-jerk limitation can be activated/deactivated by programming the setting data.

Parameter: *Value*

- Value range: TRUE, FALSE

Application:

- Part program
- Static synchronized action

4.2.14 Path jerk for real-time events (channel-specific)

4.2.14.1 General Information

Overview

So that no compromise has to be made between machining-optimized jerk on the one hand and time-optimized jerk in connection with the following real-time events on the other:

- NC Stop / NC Start
- Changing the feedrate override
- Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function

for the real-time events mentioned, the path jerk can be specified using a channel-specific system variable:

$AC_PATHJERK = *path jerk*

Path jerk for real-time events will only be active for the duration of the change in velocity in respect of one of the real-time events specified above.

Limitation

As the jerk is not a physical variable of any relevance to the drive, **no** limit is imposed on the jerk set.

Effectiveness

Effective	Path jerk for real-time events is only enabled in AUTOMATIC and MDA operating modes in conjunction with the following real-time events: • NC Stop / NC Start • Override changes • Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function
Not effective	Path jerk for real-time events is ineffective for changes in the path velocity that are attributable to path planning during preprocessing for the channel, such as contour curvatures, corners, kinematic transformation limitations, etc. Path jerk for real-time events is ineffective if the programmed value is smaller than the path jerk calculated during preprocessing for the path section concerned.

Programming

For the purpose of setting the jerk for real-time events in accordance with the acceleration, the system variables can be set as follows:

$AC_PATHJERK = $AC_PATHACC/smoothing time

- $AC_PATHACC: Path acceleration [m/s²]

  Smoothing time: Freely selectable, e.g. 0.02 s

For information about programming system variables in the part program or synchronized actions, see Section "Programming (Page 271)".

4.2.14.2 Programming

Syntax

$AC_PATHJERK = *path jerk*

Functionality

The path jerk for real-time events is set via the channel-specific system variables.

Jerk value:

- Value range: Path jerk ≥ 0
- Unit: m/s³

Application:

- Part program
- Static synchronized action

Reset response

The function is deactivated on reset.

Supplementary conditions

Programming $AC_PATHJERK in the part program automatically triggers a preprocessing stop with REORG (STOPRE).

4.2.15 Jerk with programmed rapid traverse (G00) (axis-specific)

4.2.15.1 General Information

Overview

Frequently, the maximum jerk for the machine axes involved in the machining process must be set lower than the machine's performance capability officially allows because of the supplementary conditions associated with the specific process concerned.

For time-optimized traversing of the machine axes with programmed rapid traverse (part-program instruction G00), a specific maximum value can be programmed for the axis-specific jerk.

JOG setup mode

This function does not affect jerk in respect of a rapid traverse override in JOG setup mode.

4.2.15.2 Parameterization

The maximum value for axis-specific jerk with programmed rapid traverse is parameterized (G00) using the axis-specific machine data:

MD32434 $MA_G00_ACCEL_FACTOR
(scaling of the acceleration limitation with G00)

This is used to generate the maximum value for axis-specific jerk with programmed rapid traverse (G00) that is taken into account by the path planning component during preprocessing:

Jerk[axis] =
MD32431 $MA_MAX_AX_JERK * MD32435 $MA_G00_JERK_FACTOR

4.2.16 Excessive jerk for block transitions without constant curvature (axis-specific)

4.2.16.1 General Information

Overview

In the case of block transitions without constant curvature (e.g. straight line > circle), the programmable controller has to decelerate movement of the geometry axes significantly in order to ensure compliance with the parameterized axis dynamics. For the purpose of reducing/avoiding deceleration in connection with block transitions without constant curvature, a higher level of axis-specific jerk can be enabled.

The excessive jerk is parameterized using a dedicated axis-specific maximum value.

4.2.16.2 Parameterization

The excessive jerk for block transitions without constant curvature is parameterized using the axis-specific machine data:

MD32432 $MA_PATH_TRANS_JERK_LIM
(excessive jerk for block transitions without constant curvature)

4.2.17 Velocity-dependent jerk adaptation (axis-specific)

Function

The dynamic path response results from the parameterized, constant axial maximum values for velocity, acceleration and jerk of the axes involved in the path:

- MD32000 $MA_MAX_AX_VELO (maximum axis velocity)
- MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)
- MD32431 $MA_MAX_AX_JERK (max. axial jerk for path motion)

For **contours with non-constant curvature** (torsion), as for example in connection with free-form surfaces, fluctuations in the path velocity, particularly in the upper velocity range, can result mainly due to the axial jerk. The fluctuations of the path velocity lead to adverse affects in the surface quality.

The influence of the axial jerk on the path velocity is decreased for contours with non-constant curvature through a velocity-dependent increase of the permissible axial jerk. Fluctuations in the path velocity can be avoided with the appropriate parameterization.

The velocity-dependent increase of the permissible axial jerk has no effect on the maximum possible path acceleration and path jerk. These result from the constant axial maximum values parameterized in the in the machine data even when jerk adaptation is active.

As both curvature and torsion are zero in the case of linear motion, the velocity-dependent jerk adaptation has no effect with linear motions.

Availability

The "velocity-dependent jerk adaptation" function is available independent of the function "Free-form surface mode: Basic functions (Page 222)".

Parameterization

The "Velocity-dependent jerk adaptation" function is parameterized with the following machine data:

- MD32437 $MA_AX_JERK_VEL0[<n>] = <threshold value$_{lower}$>

  Lower velocity threshold of the jerk adaptation. Velocity-dependent jerk adaptation takes effective as of this velocity.

  The lower velocity threshold can be set separately via index n for each dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 219)"):

- MD32438 $MA_AX_JERK_VEL1[<n>] = <threshold value$_{upper}$>

  Upper velocity threshold of the jerk adaptation. The velocity-dependent jerk reaches its maximum value j_{max} parameterized with MD32439 $MA_MAX_AX_JERK_FACTOR at this velocity.

  The upper velocity threshold can be set separately via index n for each dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 219)"):

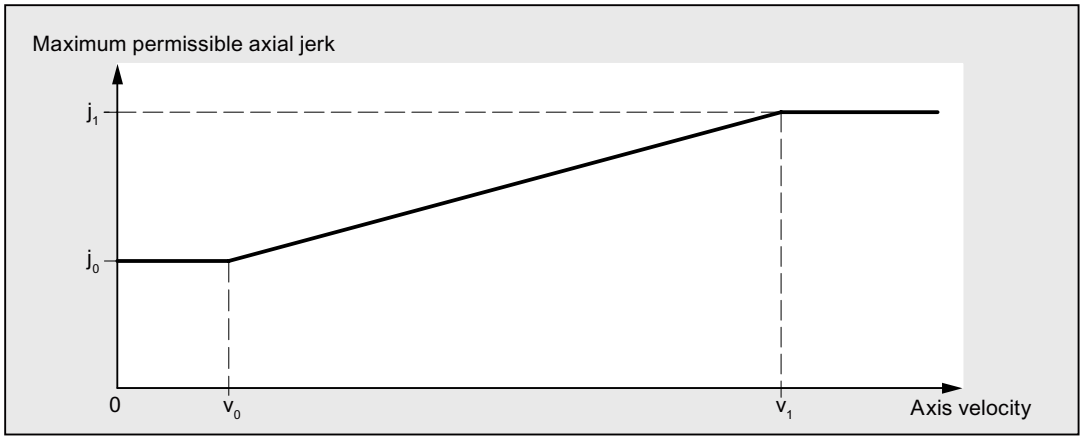
- MD32439 $MA_MAX_AX_JERK_FACTOR = <factor>

  Factor for the parameterization of the maximum velocity-dependent jerk j_{max} on reaching the upper velocity threshold MD32438 $MA_AX_JERK_VEL1[<n>]:

  j_{max} = (MD32431 $MA_MAX_AX_JERK) * (MD32439 $MA_MAX_AX_JERK_FACTOR)

  The velocity-dependent jerk adaptation is active at a value > 1.0.

  The velocity-dependent jerk adaptation is inactive at a value = 1.0.



v_0: MD32437 $MA_AX_JERK_VEL0

v_1: MD32438 $MA_AX_JERK_VEL1

j_0: MD32431 $MA_MAX_AX_JERK

j_1: MD32439 $MA_MAX_AX_JERK_FACTOR * MD32431 $MA_MAX_AX_JERK

Figure 4-5 Axial jerk as a function of the axis velocity

Note

The velocity-dependent jerk adaptation is only active, if:

MD32439 $MA_MAX_AX_JERK_FACTOR > 1.0

Example

Example of parameter assignment:

- MD32437 $MA_AX_JERK_VEL0 = 3000 mm/min
- MD32438 $MA_AX_JERK_VEL1 = 6000 mm/min
- MD32439 $MA_MAX_AX_JERK_FACTOR[AX**1**] = 2.0
- MD32439 $MA_MAX_AX_JERK_FACTOR[AX**2**] = 3.0
- MD32439 $MA_MAX_AX_JERK_FACTOR[AX**3**] = 1.0

Effect

- The velocity-dependent jerk adaptation becomes active for the 1st and 2nd axis, while the function for the 3rd axis is not active.
- The parameterized jerk is effective at axis velocities in the range of 0 to 3000 mm/min.
- The maximum jerk is linearly increased for axis velocities in the range 3000 mm/min to 6000 mm/min.
- The maximum permitted jerk of the 1st axis is, for axis velocities greater than 6000 mm/min, increased by a factor of 2 - for the 2nd axis, by factor of 3.
- The parameterized values apply in each dynamic response mode.

4.2.18 Jerk filter (axis-specific)

4.2.18.1 General Information

Overview

In certain application scenarios, e.g. when milling free-form surfaces, it may be beneficial to smooth the position setpoint characteristics of the machine axes. This enables surface quality to be improved by reducing the mechanical vibrations generated in the machine.

For the purpose of smoothing the position setpoint characteristic of a machine axis, a jerk filter can be activated at position controller level, independently of the channel- and axis-specific jerk limitations taken into account at interpolator level.

The effect of the jerk filter must be as strong as possible without having an unacceptable impact on contour accuracy. The filter should also have as "balanced" a smoothing effect as possible, i.e. if the same contour is traversed forwards and backwards, the contour smoothed by the filter should be as similar as possible in both directions.

To enable the jerk filter to be optimally matched to the machine conditions, various filter modes are available:

- 2nd-order filter (PT2)
- Sliding mean value generation
- Bandstop filter

Mode: 2nd-order filter

Owing to the fact that it is a simple low-pass filter, "2nd-order filter" mode can only meet the requirements specified above where relatively small filter time constants (around 10 ms) are concerned. When used in conjunction with larger time constants, impermissible contour deviations are soon manifest. The effect of the filter is relatively limited.

This filter mode offers advantages if very large filter time constants are needed and contour accuracy is only of secondary importance (e.g. positioning axes).

For historical reasons, this filter mode is set as the default.

Mode: Sliding mean value generation

Where minimal contour deviations are required, filter time constants within the range of 20-40 ms can be set using the "sliding mean value generation" filter mode. The smoothing effect is largely symmetrical.

The display of the calculated servo gain factor (KV factor) in the user interface, shows smaller values than would normally be expected for the filter. The contour accuracy is in fact higher than the displayed KV filter appears to suggest.

When changing from "2nd-order filter" to "sliding mean value generation" filter mode, the displayed KV factor may, therefore, drop (with identical filter time constant), even though there is an improvement in contour accuracy.

Mode: Bandstop filter

The bandstop filter is a 2nd-order filter in terms of numerator and denominator:

$$H(s) = \frac{\frac{s^2}{(2 * \pi * f_Z)^2} + \frac{2 * s * D_Z}{(2 * \pi * f_Z)}}{\frac{s^2}{(2 * \pi * f_N)^2} + \frac{2 * s * D_N}{2 * \pi * f_N}}$$

where:

f_Z: Numerator natural freq.

f_N: Denominator natural freq.

D_Z: Numerator damping

D_N: Denominator damping

Since a vibration-capable filter setting is not expected to yield useful results in any case, as with the jerk filter's "2nd-order filter" (PT2) low-pass filter (PT2) mode there is no setting option for the denominator damping D_N. The denominator damping D_N is permanently set to 1.

The bandstop filter can be parameterized in two different ways:

- Real bandstop filter
- Bandstop filter with additional amplitude response increase/decrease at high frequencies

Real bandstop filter

The real bandstop filter is applied when identical numerator and denominator natural frequencies are selected:

$f_Z = f_N = f_{block}$ (blocking frequency)

If numerator damping setting = 0 is selected, the blocking frequency is equivalent to complete attenuation. In this case the 3 dB bandwidth is calculated as follows:

$f_{3\ dB\ bandwidth} = 2 * f_{block}$

If instead of complete attenuation, a reduction by a factor of k is all that is required, then numerator damping should be selected in accordance with k. In this case the above formula for calculating the 3 dB bandwidth no longer applies.

Bandstop filter with additional amplitude response increase/decrease at high frequencies

In this case, the numerator and denominator natural frequencies are set to different values. The numerator natural frequency determines the blocking frequency.

By selecting a lower/higher denominator natural frequency than the numerator natural frequency, you can increase/decrease the amplitude response at high frequencies. An amplitude response increase at high frequencies can be justified in most cases, as the controlled system generally possesses a lowpass characteristic itself, i.e. the amplitude response drops at high frequencies anyway.

Supplementary conditions

If too high a numerator natural frequency is selected, the filter is disabled. In this case the limiting frequency f_{Zmax} depends on the position-control cycle:

$$f_{Zmax} = \frac{1}{2 * \pi * T_{Zmin}} = \frac{1}{2 * \pi * T_{\text{Position-control cycle}}} \quad \text{(Shannon Theorem)}$$

4.2.18.2 Parameterization

Activation

The jerk filter is activated using the machine data:

MD32400 $MA_AX_JERK_ENABLE (axial jerk limitation)

The jerk filter is active in all operating modes and with all types of interpolation.

Filter mode

The filter mode is selected via the machine data:

MD32402 $MA_AX_JERK_MODE

Value	Filter mode
1	2nd-order filter
2	Sliding mean value generation
3	Bandstop filter

Time constant

The time constant for the axial jerk filter is set with the machine data:

MD32410 $MA_AX_JERK_TIME

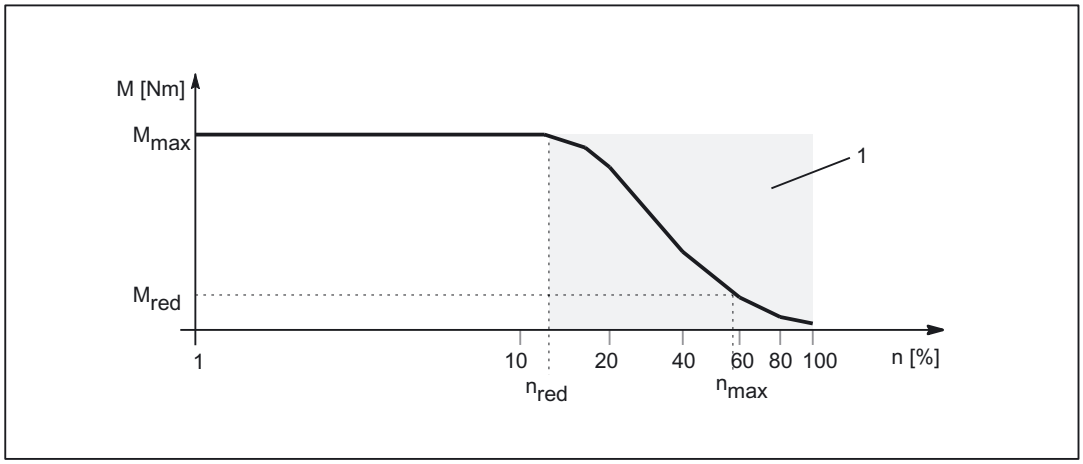
The jerk filter is only effective when the time constant is greater than a position-control cycle.

4.2.19 Kneeshaped acceleration characteristic curve

4.2.19.1 Adaptation to the motor characteristic curve

Function

Various types of motor, particularly stepper motors, have a torque characteristic that is highly dependent upon speed and shows a steep decrease in torque in the upper speed range. To ensure optimum utilization of the motor characteristic curve, it is necessary to reduce the acceleration once a certain speed is reached.



1: Torque decrease zone

n_{red}: Speed above which reduced torque has to be assumed

n_{max}: Maximum speed

M_{max}: Max. torque

M_{red}: Torque at n_{max} (corresponds to creep acceleration)

Figure 4-6 Torque characteristic curve of a motor with torque characteristic that is highly dependent upon speed

Simulation of torque characteristic

For the purpose of simulating the torque characteristic of the motor characteristic curve, the machine data:
MD35242 $MA_ACCEL_REDUCTION_TYPE = *characteristic*
can be used to select various types of characteristic curve:

0	= Constant characteristic
1	= Hyperbolic characteristic
2	= Linear characteristic

The following figures show typical velocity and acceleration characteristic curves for the respective types of characteristic:

Constant characteristic

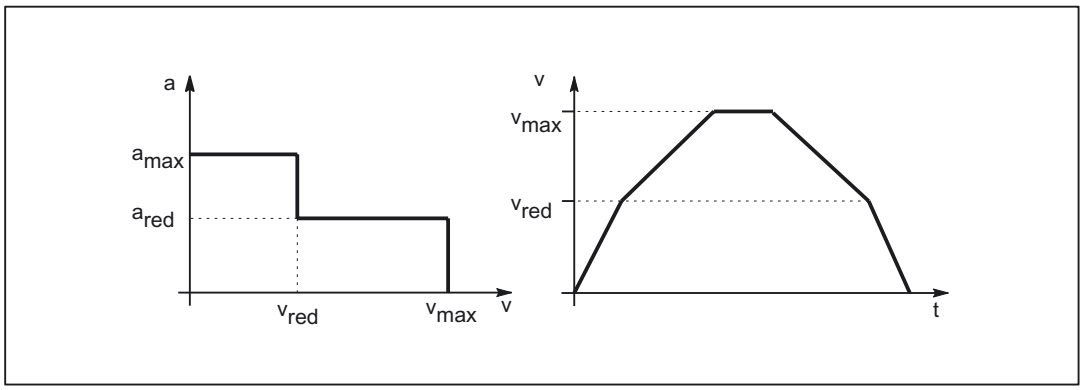


Figure 4-7 Acceleration and velocity characteristic with acceleration reduction: 0 = constant

Hyperbolic characteristic

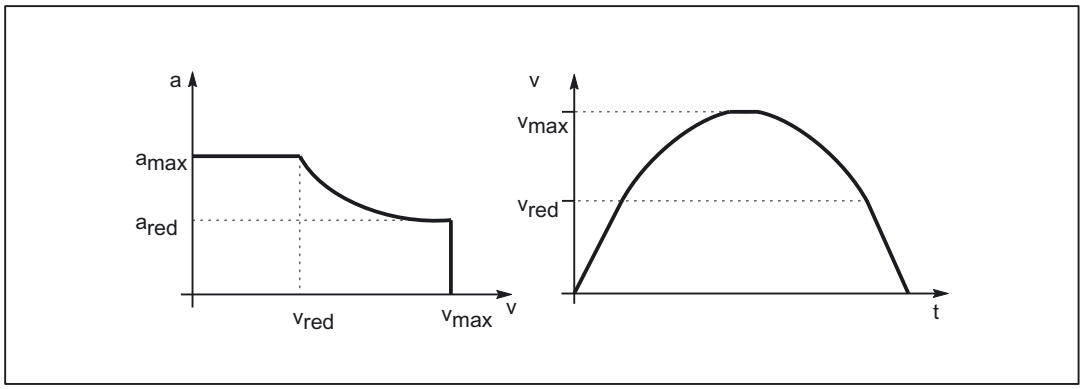


Figure 4-8 Acceleration and velocity characteristic with acceleration reduction: 1 = hyperbolic

Linear characteristic

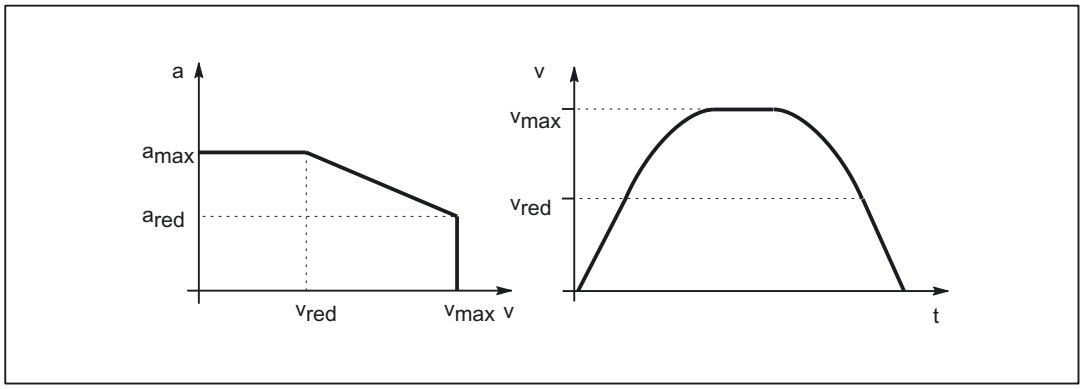


Figure 4-9 Acceleration and velocity characteristic with acceleration reduction: 2 = linear

The key data for the characteristic curves equate to:

v_{max} = $MA_MAX_AX_VELO
v_{red} = $MA_ACCEL_REDUCTION_SPEED_POINT * $MA_MAX_AX_VELO
a_{max} = $MA_MAX_AX_ACCEL
a_{red} = (1 - $MA_ACCEL_REDUCTION_FACTOR) * $MA_MAX_AX_ACCEL

4.2.19.2 Effects on path acceleration

Function

The path acceleration characteristic curve is generated on the basis of the types of characteristic for the axes that are of relevance for the path. If axes with different types of characteristic curve are interpolated together, the acceleration profile for the path acceleration will be determined on the basis of the reduction type that is most restrictive.

The following order of priorities applies, whereby 1 = top priority:

1. Acceleration reduction: 0 = constant characteristic
2. Acceleration reduction: 1 = hyperbolic characteristic
3. Acceleration reduction: 2 = linear characteristic
4. No acceleration reduction effective

   A situation, whereby no acceleration reduction is active, arises for example when:

   MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT = 1

   and/or

   MD35230 $MA_ACCEL_REDUCTION_FACTOR = 0

Note

Machine axes featuring stepper motor and DC drive can be interpolated together.

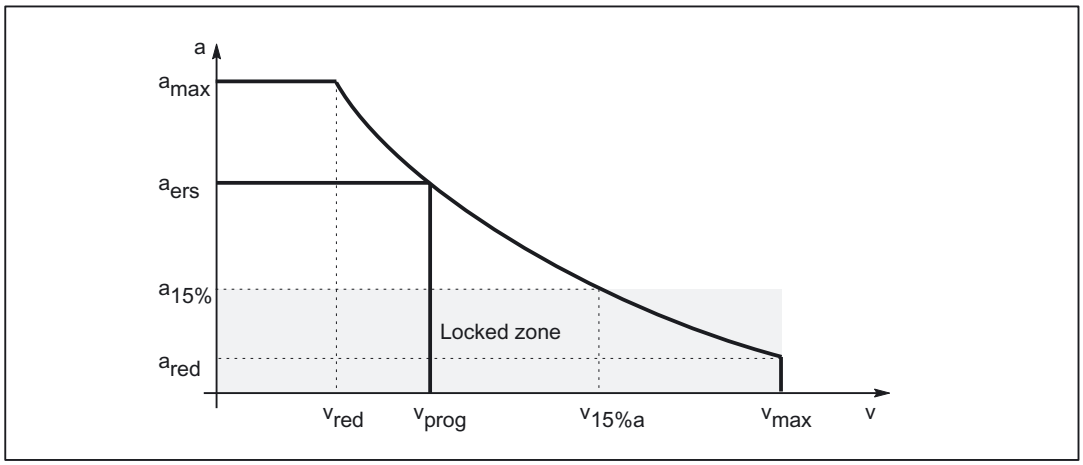
4.2.19.3 Substitute characteristic curve

Function

If the programmed path cannot be traversed using the parameterized acceleration characteristic curve (e.g., active kinematic transformation), a substitute characteristic curve is generated by reducing the dynamic limit values. The dynamic limit values are calculated to ensure that the substitute characteristic curve provides the best possible compromise between maximum velocity and constant acceleration.

Substitute characteristic curve with linear path sections

Limitation to this value is applied if the programmed path velocity is greater than that at which 15 % of the maximum acceleration capacity is still available ($v_{15\%a}$). Consequently, 15 % of the maximum acceleration capacity/motor torque always remains available, whatever the machining situation.



a_{ers}: Substitute characteristic curve constant acceleration

$a_{15\%}$: Minimal constant acceleration

$a_{15\%} = 0.15 * (a_{max} - a_{red}) + a_{red}$

v_{ers}: Substitute characteristic curve velocity

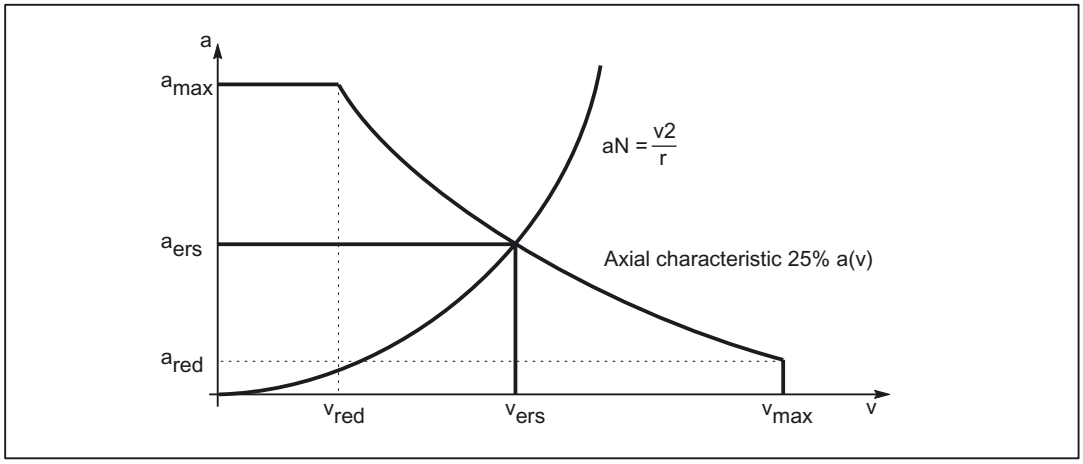
v_{prog}: Programmed velocity

$v_{15\%a}$: Velocity at $a_{15\%}$

Figure 4-10 Substitute path characteristic curve: Linear path

Substitute characteristic curve with curved path sections

In the case of curved path sections, normal and tangential acceleration are considered together. The path velocity is reduced so that only up to 25 % of the speed-dependent acceleration capacity of the axes is required for normal acceleration. The remaining 75 % of the acceleration capacity is set aside for the tangential acceleration, i.e., deceleration/acceleration on the path.



a_N: Normal acceleration

a_{ers}: Substitute characteristic curve constant acceleration

v_{ers}: Substitute characteristic curve velocity

r: Path radius

Figure 4-11 Substitute path characteristic curve: Curved path

Block transitions with continuous-path mode

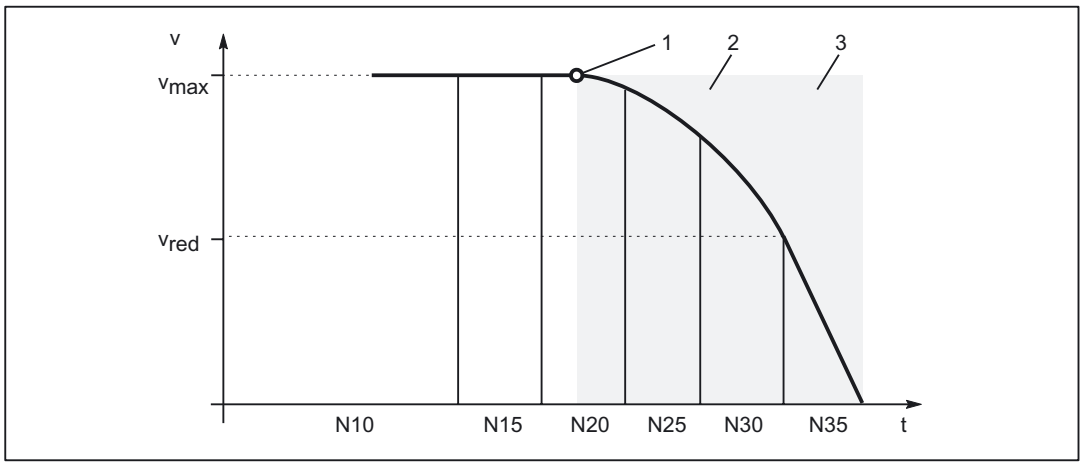
If continuous-path mode is active, non-tangential block transitions result in axial velocity jumps when the programmed path velocity is used for traversing.

As a result, the path velocity is controlled in such a way that prevents any axial velocity proportion from exceeding the creep velocity v_{red} at the time of the block transition.

Deceleration ramp with continuous-path mode and LookAhead

In the case of consecutive part program blocks with short paths, an acceleration or deceleration operation may be spread over several part program blocks.

In such a situation LookAhead also takes into account the parameterized speed-dependent acceleration characteristic.



1: Brake application point
2: Torque decrease zone
3: Maximum torque zone
v_{red}: Creep velocity
v_{max}: Maximum velocity
Nxy: Part program block with block number Nxy

Figure 4-12 Deceleration with LookAhead

4.2.19.4 Parameterization

The knee-shaped acceleration characteristics can be activated specific to the machine axis via the machine data:

MD35240 $MA_ACCEL_TYPE_DRIVE = TRUE

The knee-shaped acceleration characteristic curve is parameterized for specific axes using machine data:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT
(speed for reduced acceleration)

MD35230 $MA_ACCEL_REDUCTION_FACTOR (reduced acceleration)

MD32300 $MA_MAX_AX_ACCEL (Maximum axis acceleration)

MD35242 $MA_ACCEL_REDUCTION_TYPE
(type of acceleration reduction: 0 = constant, 1 = hyperbolic, 2 = linear)

4.2.19.5 Programming

Channel-specific activation (DRIVE)

Syntax

```
DRIVE
```

Functionality

The knee-shaped characteristic curve is activated for path acceleration using the `DRIVE` part-program instruction.

G group: 21

Effective: Modal

Reset response

The channel-specific default setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

Dependencies

If the knee-shaped acceleration characteristic curve is parameterized for a machine axis, then this becomes the default acceleration profile for all traversing operations.

If the effective acceleration profile is changed for a specific path section using the `SOFT` or `BRISK` part-program instructions, then an appropriate substitute characteristic curve with lower dynamic limit values is used in place of the knee-shaped acceleration characteristic curve.

The knee-shaped acceleration characteristic curve can be reactivated by reprogramming `DRIVE`.

Axis-specific activation (DRIVEA)

Syntax

`DRIVEA` (*Axis* {*Axis*})

Functionality

The knee-shaped characteristic curve is activated for all single-axis interpolations (positioning axis, reciprocating axis, etc.) for specific axes using the part-program instruction.

G group: -

Effective: modal

Axis:

- Value range: Axis identifier for channel axes

Reset response

The channel-specific default setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

Dependencies

If the knee-shaped acceleration characteristic curve is parameterized for a machine axis, then this becomes the default acceleration profile for all traversing operations.

If the effective acceleration profile is changed for a specific axis using the SOFTA or BRISKA part-program instructions, then an appropriate substitute characteristic curve is used in place of the knee-shaped acceleration characteristic curve.

It is possible to switch back to the knee-shaped acceleration characteristic curve for a specific axis by programming DRIVEA.

4.2.19.6 Boundary conditions

Single axis interpolation

After activating the knee-shaped acceleration characteristics in case of single-axis interpolations (positioning axis, oscillating axis, manual motion, etc.), the machine axis is traversed exclusively in the mode DRIVEA .

It is not possible to switch over the acceleration profile via the following part program instructions:

- Abrupt acceleration changes (BRISKA)
- Acceleration with jerk limitation (SOFTA)

Path interpolation

If for a machine axis involved in a programmed path the knee-shaped acceleration characteristic parameterized without the part program instruction DRIVE is active, then a substitute characteristic curve with reduced dynamic limiting values is determined for the path.

Kinematic transformation

The knee-shaped acceleration characteristic is not considered in an active kinematic transformation. With internal control, a switchover is done to acceleration without jerk limitation (BRISK) and a substitute characteristic curve becomes active for the path acceleration.

4.2.20 Acceleration and jerk for JOG motions

In order to avoid unwanted machine motions in JOG mode, separate axial acceleration and jerk limit values can be specified for JOG motion.

It is also possible to limit acceleration and jerk channel-specifically for manual traversing of geometry and orientation axes. This enables better handling of the kinematics that generate Cartesian motions entirely via rotary axes (robots).

4.2.20.1 Parameterization

Axial limitation of acceleration and jerk

Maximum axial acceleration for JOG motions

The maximum axial acceleration for JOG motion can be specified for each machine axis via the machine data:

MD32301 $MA_JOG_MAX_ACCEL

With MD32301 = 0, the value from MD32300 $MA_MAX_AX_ACCEL is effective instead of the JOG-specific maximum value.

Maximum axial jerk for JOG motions

The maximum axial jerk for JOG motion can be specified for each machine axis via the machine data:

MD32436 $MA_JOG_MAX_JERK

With MD32436 = 0, the value from MD32430 $MA_JOG_AND_POS_MAX_JERK is effective instead of the JOG-specific maximum value.

Note

MD32436 $MA_JOG_MAX_JERK is only effective when the axial jerk limitation in JOG mode has been enabled for the machine axes to be traversed:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE

This is also possible by programming `SOFTA(<Axis1>,<Axis2>, ... )` in the part program.

Channel-specific limitation of acceleration and jerk

Maximum acceleration when manually traversing geometry axes

The maximum acceleration when manually traversing geometry axes can be specified for each channel via the machine data:

MD21166 $MC_JOG_ACCEL_GEO [<geometry axis>]

With <geometry axis> = 0, 1, 2

With MD21166 = 0, the axis-specific limit value from MD32301 $MA_JOG_MAX_ACCEL is effective instead of the channel-specific acceleration limitation.

Note

With MD21166 $MC_JOG_ACCEL_GEO [<geometry axis>], there is no direct limitation to MD32300 $MA_MAX_AX_ACCEL.

Note

When a transformation is active, MD32300 $MA_MAX_AX_ACCEL determines the maximum possible axial acceleration.

Maximum jerk when manually traversing geometry axes

The maximum jerk when manually traversing geometry axes in the SOFT acceleration mode (acceleration with jerk limitation) can be specified for each channel via the machine data:

MD21168 $MC_JOG_JERK_GEO [<geometry axis>]

With <geometry axis> = 0, 1, 2

With MD21168 = 0, the axis-specific limit value from MD32436 $MA_JOG_MAX_JERK is effective instead of the channel-specific jerk limitation.

Note

MD21168 $MC_JOG_JERK_GEO is only effective when the axial jerk limitation in JOG mode has been enabled for the basic machine axes:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE

Maximum jerk when manually traversing orientation axes

The maximum jerk when manually traversing orientation axes can be specified for each channel via the machine data:

MD21158 $MC_JOG_JERK_ORI [<orientation axis>]

For MD21158 to take effect, the channel-specific jerk limitation for the manual traversing of orientation axes must be enabled via the following machine data:

MD21159 $MC_JOG_JERK_ORI_ENABLE == TRUE

Note

Orientation axes are not affected by the machine data MD32301 $MA_JOG_MAX_ACCEL and MD32436 $MA_JOG_MAX_JERK.

4.2.20.2 Supplementary conditions

Path override / overlaid motion

With path override / overlaid motion (e.g. DRF), the JOG-specific maximum values for acceleration and jerk (MD32301 $MA_JOG_MAX_ACCEL and MD32436 $MA_JOG_MAX_JERK) are **not** effective. Instead, the values for positioning axis motions are effective:

- Acceleration:
  - MD32300 $MA_MAX_AX_ACCEL [0] (maximum axial acceleration for path motions in the dynamic response mode DYNNORM)

    See also Section "Acceleration without jerk limitation (BRISK/BRISKA) (channel/axis-specific) (Page 247)".
- Jerk
  - MD32430 $MA_JOG_AND_POS_MAX_JERK (maximum axial jerk for positioning axis motions)

    or (with G75/751):
  - MD32431 $MA_MAX_AX_JERK [0] (maximum axial jerk for path motions in the dynamic response mode DYNNORM)

  See also Section "Jerk limitation with single-axis interpolation (SOFTA) (axis-specific) (Page 266)".

Note

Only the dynamic response mode DYNNORM is always effective for JOG mode.

Behavior for manual traversing of geometry axes with active rotation

When manually traversing geometry axes in the SOFT acceleration mode (acceleration with jerk limitation), the value from MD32436 $MA_JOG_MAX_JERK or MD32430 $MA_JOG_AND_POS_MAX_JERK is also used with active rotation or active orientable toolholder.

Part program instruction SOFTA/BRISKA/DRIVEA

The part program instruction SOFTA(<Axis1>,<Axis2>, ...) is also effective in JOG mode, i.e. the maximum axial jerk from MD32436 $MA_JOG_MAX_JERK is effective for the specified axes when traversing in JOG mode (exactly as when setting MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE).

Note

On the other hand the SOFT part program instruction has no effect on the JOG mode.

As with SOFTA, the part program instructions BRISKA and DRIVEA are also effective in JOG mode, i.e. the acceleration is without jerk limitation, even when MD32420 $MA_JOG_AND_POS_JERK_ENABLE is set to "TRUE" for the relevant machine axes.

Note

The manual traversing of orientation axes is not affected by BRISKA/SOFTA/DRIVEA.

4.3 Examples

4.3.1 Acceleration

4.3.1.1 Path velocity characteristic

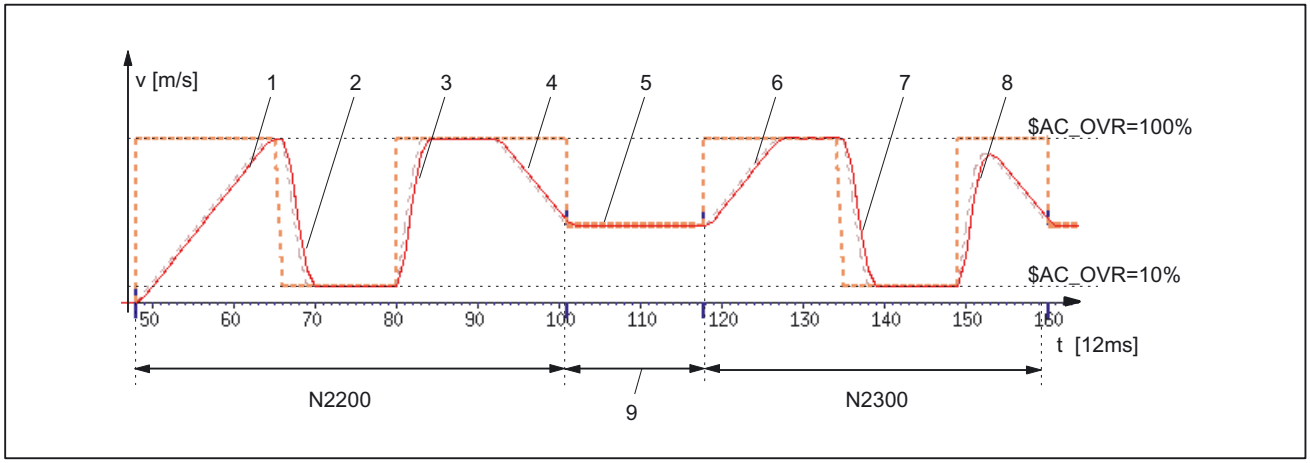
Key statement

An excerpt from a part program is provided below, together with the associated acceleration characteristic, by way of an example. These are used to illustrate how the path velocity can be adapted to take account of various events and the resulting change in acceleration.

Part program (excerpt, schematic)

Program code

```
; Acceleration selection in accordance with fast input 1 ($A_IN[1]):
N53 ID=1 WHEN $A_IN[1] == 1 DO $AC_PATHACC = 2.*$MA_MAX_AX_ACCEL[X]

; Test override profile (simulates external intervention):
N54 ID=2 WHENEVER ($AC_TIMEC > 16) DO $AC_OVR=10
N55 ID=3 WHENEVER ($AC_TIMEC > 30) DO $AC_OVR=100

;Approach
N1000 G0 X0 Y0 BRISK
N1100 TRANS Y=-50
N1200 AROT Z=30 G642

; Contour
N2100 X0 Y0
N2200 X = 70 G1 F10000 RNDM=10 ACC[X]=30 ACC[Y]=30
N2300 Y = 70
N2400 X0
N2500 Y0
M30
```



Acceleration profile: BRISK

1: Accelerate to 100% of path velocity (F10000) in accordance with acceleration default: ACC (N2200...)

2: Brake to 10% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N54...)

3: Accelerate to 100% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N55...)

4: Brake to block end velocity for intermediate smoothing block in accordance with acceleration default: ACC (N2200...)

5: Speed limitation as a result of smoothing (see 9)

6: Accelerate to 100% of path velocity ($AC_OVR) in accordance with acceleration default: ACC (N2300...)

7: Decelerate as a result of override modification at a rate of acceleration that is in accordance with real-time acceleration $AC_PATHACC (N53/N54...)

8: Accelerate to 100% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N55...)

9: Intermediate block inserted within the control as a result of the programmed smoothing (RNDM) (N2200...)

Figure 4-13 Switching between path acceleration specified during preprocessing and real-time acceleration

4.3.2 Jerk

4.3.2.1 Path velocity characteristic

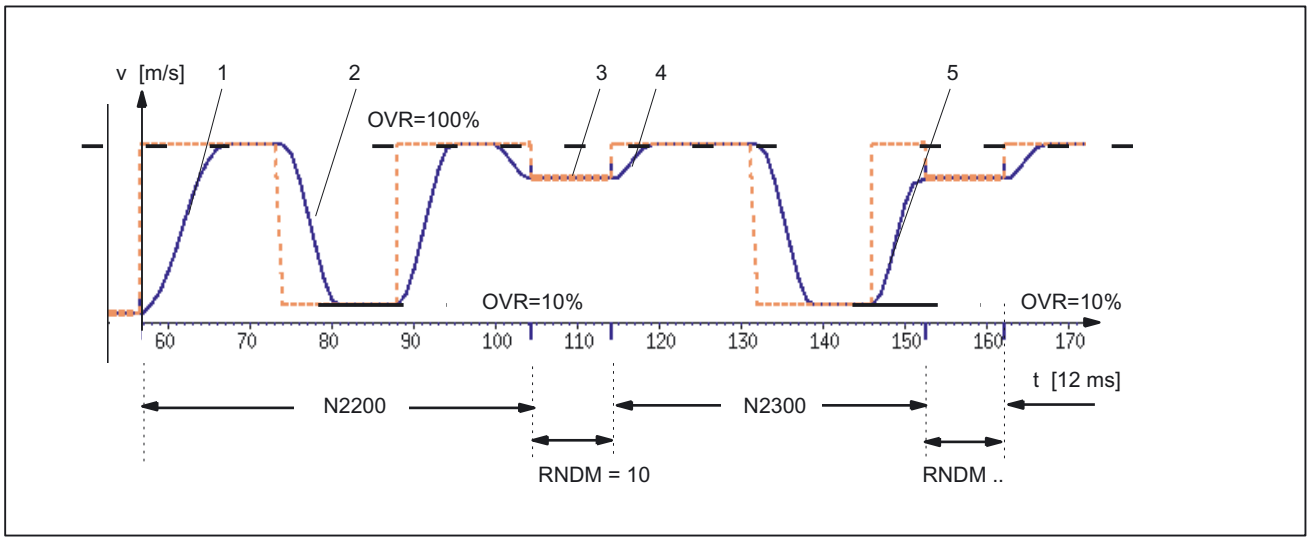
Key statement

An excerpt from a part program is provided below, together with the associated acceleration characteristic, by way of an example. These are used to illustrate how the path velocity can be adapted to take account of various events and the resulting change in jerk.

Part program (excerpt, schematic)

Program code

```
; Setting of path acceleration and path jerk in the event of external intervention:
N0100 $AC_PATHACC = 0.
N0200 $AC_PATHJERK = 4. * ($MA_MAX_AX_JERK[X] + $MA_MAX_AX_JERK[Y]) / 2.

; Synchronized actions for the purpose of varying the override (simulates external
intervention):
N53 ID=1 WHENEVER ($AC_TIMEC > 16) DO $AC_OVR=10
N54 ID=2 WHENEVER ($AC_TIMEC > 30) DO $AC_OVR=100

;Approach
N1000 G0 X0 Y0 SOFT
N1100 TRANS Y=-50
N1200 AROT Z=30 G642

; Contour
N2100 X0 Y0
N2200 X = 70 G1 F10000 RNDM=10
N2300 Y = 70
N2400 X0
N2500 Y0
M30
```



Acceleration profile: SOFT

1: Jerk according to $MA_MAX_AX_JERK[..]
2: Jerk according to $AC_PATHJERK
3: Jerk according to $MA_MAX_AX_JERK[..] (approach block end velocity)
4: Velocity limit due to arc
5: Jerk according to $AC_PATHJERK

Figure 4-14 Switching between path jerk specified during preprocessing and $AC_PATHJERK

4.3.3 Acceleration and jerk

Key statement

In the following example a short part program is used to illustrate the velocity and acceleration characteristic for the X-axis. It also shows the connection between specific velocity and acceleration-related machine data and the contour sections they influence.

Part program

Program code	Comment
N90 F5000 SOFT G64	; Continuous-path mode, jerk-limited acceleration
N100 G0 X0 Y0 Z0	; Rapid traverse
N110 G1 X10	; Straight line
N120 G3 CR=5 X15 Y5	; Circular arc, radius 5 mm, block transition: Tangential
N130 G3 CR=10 X5 Y15	; Circular arc, radius 10 mm, block transition: Tangential
N140 G1 X-5 Y17.679	; Straight line, 15° bend
N200 M30	

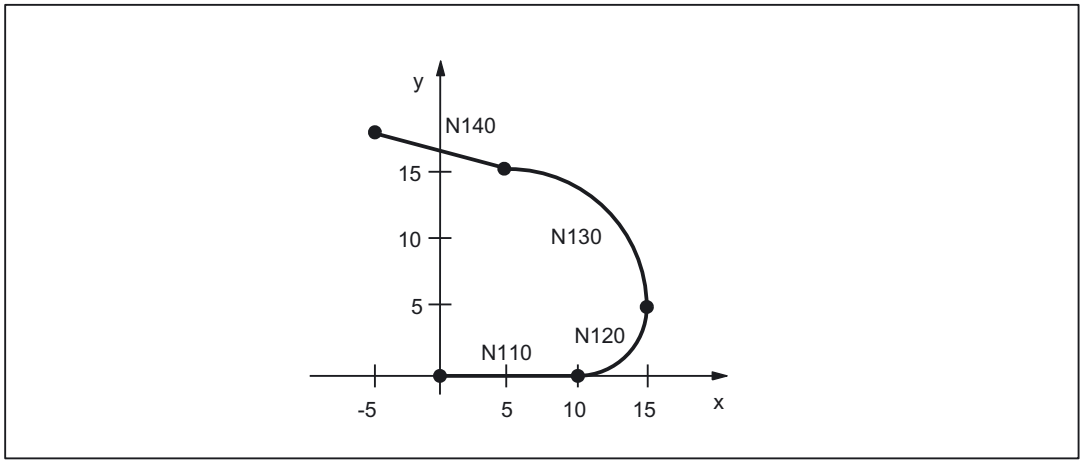


Figure 4-15 Part program contour

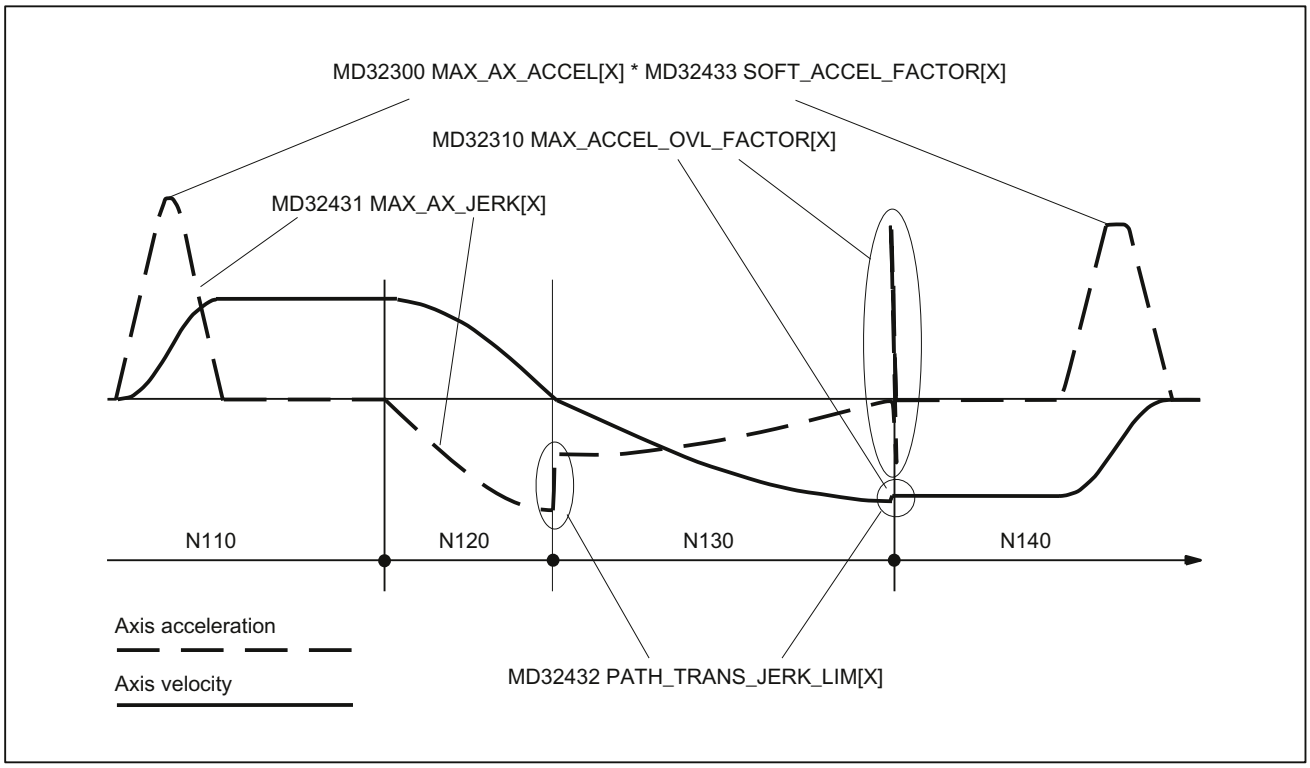


Figure 4-16 X axis: Velocity and acceleration characteristic

4.3.4 Knee-shaped acceleration characteristic curve

4.3.4.1 Activation

Key statement

The example given illustrates how the knee-shaped acceleration characteristic curve is activated on the basis of:

- Machine data
- Part program instruction

Machine data

- Parameterizing the characteristic curve (example only)

```
Program code
X axis
MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT[X] = 0.4
MD35230 $MA_ACCEL_REDUCTION_FACTOR[X] = 0.85
MD35242 $MA_ACCEL_REDUCTION_TYPE[X] = 2
MD35240 $MA_ACCEL_TYPE_DRIVE[X] = TRUE

Y axis
MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT[Y] = 0.0
MD35230 $MA_ACCEL_REDUCTION_FACTOR[Y] = 0.6
MD35242 $MA_ACCEL_REDUCTION_TYPE[Y] = 1
MD35240 $MA_ACCEL_TYPE_DRIVE[Y] = TRUE

Z axis
MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT[Z] = 0.6
MD35230 $MA_ACCEL_REDUCTION_FACTOR[Z] = 0.4
MD35242 $MA_ACCEL_REDUCTION_TYPE[Z] = 0
MD35240 $MA_ACCEL_TYPE_DRIVE[Z] = FALSE
```

- Activation by setting as the channel-specific default setting
  MC_GCODE_RESET_VALUE[20] = 3 (DRIVE)

Part program (excerpt)

```
Program code                          Comment
N10 G1 X100 Y50 Z50 F700              ; Path motion (X,Y, Z) with DRIVE
N15 Z20                               ; Path motion (Z) with DRIVE
N20 BRISK                             ; Switchover to BRISK
N25 G1 X120 Y70                       ; Path motion (Y, Z) with substitute
                                        characteristic curve
N30 Z100                              ; Path motion (Z) with BRISK
N35 POS[X] = 200 FA[X] = 500          ; Positioning motion (X) with DRIVEA
N40 BRISKA(Z)                         ; Activate BRISKA for Z
N40 POS[Z] = 50 FA[Z] = 200           ; Positioning motion (Z) with BRISKA
N45 DRIVEA(Z)                         ; Activate DRIVEA for Z
N50 POS[Z] = 100                      ; Positioning motion (Z) with DRIVE
N55 BRISKA(X)                         ; results in error message
...
```

4.4 Data lists

4.4.1 Machine data

4.4.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
18960	POS_DYN_MODE	Type of positioning axis dynamic response

4.4.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Initial setting of the G groups
20500	CONST_VELO_MIN_TIME	Minimum time with constant velocity
20600	MAX_PATH_JERK	Path-related maximum jerk
20602	CURV_EFFECT_ON_PATH_ACCEL	Influence of path curvature on path dynamic response
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid motions
21158	JOG_JERK_ORI	Maximum jerk of the orientation axes when traversing in JOG
21159	JOG_JERK_ORI_ENABLE	Initial setting of the channel-specific jerk limitation of orientation axes for traversing in JOG mode
21166	JOG_ACCEL_GEO	Maximum acceleration rate of the geometry axes when traversing in JOG
21168	JOG_JERK_GEO	Maximum jerk of the geometry axes when traversing in JOG

4.4.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32000	MAX_AX_VELO	Maximum axis velocity
32300	MAX_AX_ACCEL	Maximum axis acceleration
32301	JOG_MAX_ACCEL	Maximum acceleration in JOG mode
32310	MAX_ACCEL_OVL_FACTOR	Overload factor for velocity jump
32320	DYN_LIMIT_RESET_MASK	Reset behavior of dynamic limits
32400	AX_JERK_ENABLE	Axial jerk limitation
32402	AX_JERK_MODE	Filter type for axial jerk limitation
32410	AX_JERK_TIME	Time constant for axial jerk filter
32420	JOG_AND_POS_JERK_ENABLE	Initial setting for axial jerk limitation
32430	JOG_AND_POS_MAX_JERK	Axial jerk for single axis motion
32431	MAX_AX_JERK	Maximum axial jerk at the block transition in continuous-path mode
32432	PATH_TRANS_JERK_LIM	Max. axial jerk of a geometry axis at block boundary
32433	SOFT_ACCEL_FACTOR	Scaling of acceleration limitation for `SOFT`
32434	G00_ACCEL_FACTOR	Scaling of acceleration limitation for `G00`
32435	G00_JERK_FACTOR	Scaling of axial jerk limitation for `G00`
32436	JOG_MAX_JERK	Maximum axial jerk for JOG motion
32437	AX_JERK_VEL0	First velocity threshold for velocity-dependent jerk adaptation
32438	AX_JERK_VEL1	Second velocity threshold for the velocity-dependent jerk adaptation
32439	MAX_AX_JERK_FACTOR	Factor to set the maximum jerk for higher velocities (velocity-dependent jerk adaptation)
35220	ACCEL_REDUCTION_SPEED_POINT	Speed for reduced acceleration
35230	ACCEL_REDUCTION_FACTOR	Reduced acceleration
35240	ACCEL_TYPE_DRIVE	DRIVE acceleration characteristic for axes on/off
35242	ACCEL_REDUCTION_TYPE	Type of acceleration reduction

4.4.2 Setting data

4.4.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42500	SD_MAX_PATH_ACCEL	Max. path acceleration
42502	IS_SD_MAX_PATH_ACCEL	Analysis of SD 42500: ON/OFF
42510	SD_MAX_PATH_JERK	Max. path-related jerk
42512	IS_SD_MAX_PATH_JERK	Analysis of SD 42510: ON/OFF

4.4.3 System variables

Identifier	Description
$AC_PATHACC	Path acceleration for real-time events
$AC_PATHJERK	Path jerk for real-time events

F1: Travel to fixed stop

5

5.1 Brief description

Function

With the "Travel to fixed stop" function, moving machine parts, e.g. tailstock or sleeve, can be traversed with a defined torque against other machine parts.

Characteristics

- The clamping torque and a fixed stop monitoring window can be programmed in the part program and can also be altered via setting data once the fixed stop has been reached.
- Travel to fixed stop is possible for axes and spindles.
- Travel to fixed stop is possible simultaneously for several axes and parallel to the traversing of other axes.
- The reduction of the clamping torque or clamping force can be switched on and off in synchronized actions.
- The "Travel to fixed stop" functions can also be activated via synchronized actions.
- A multi-channel block search can be performed with calculation of all the required additional data (SERUPRO).
- The axes can be simulated with FXS and FOC.
- "Vertical" axes can also be moved with FXS to a fixed stop.

5.2 General functionality

5.2.1 Programming

Function

Travel to fixed stop

The "Travel to fixed stop" function is controlled via the FXS, FXST and FXSW commands.

The activation can also be performed without traversing motion of the relevant axis. The torque is immediately limited. The fixed stop is monitored as soon as the axis is traversed.

Note

Synchronized actions

The "Travel to fixed stop" function can also be controlled via synchronized actions.

References:
Function Manual, Synchronized Actions

Travel with limited torque/force

Travel with limited torque/force can be controlled via the FOCON, FOCOF and FOC commands (see Section "Travel with limited torque/force FOC (Page 323)").

Syntax

```
FXS[<axis>]=<request>
FXST[<axis>]=<clamping torque>
FXSW[<axis>] = <window width>
FOCON[<axis>]
FOCOF[<axis>]
FOC[<axis>]
```

Meaning

Parameters	Meaning
FXS:	Travel to fixed stop Effectiveness: Modal
<Request>:	Request to the "Travel to fixed stop" function: 0 = switch off 1 = switch on
FXST:	Set clamping torque
<Clamping torque>:	Clamping torque in % of the maximum drive torque. SINAMICS S120: p2003
FXSW:	Set monitoring window
<Window width>:	Width of the tolerance window around the fixed stop Unit: mm, inch or degrees
FOCON:	Torque/force limitation: Switch on
FOCOF:	Torque/force limitation: Switch off
FOC:	Non-modal torque/force limitation
<axis>:	Identifier of the channel axis Type: AXIS

Axis identifier

Instead of a machine axis identifier, the identifiers of geometry or special axes can be used if they are assigned precisely to a machine axis and one of the following functions is not active for the machine axis:

- Transformation
- Coupling
- Frame

Change clamping torque

The clamping torque can be changed with the FXST command. The change takes effect before the traversing motions of the block. The torque limitation acts in addition to the acceleration limitation (ACC).

Ramp-shaped change

A time within which the clamping torque should be changed can be set via the following machine data. The change of the clamping torque is then no longer sudden, but ramp-shaped.

MD37012 $MA_FIXED_STOP_TORQUE_RAMP_TIME (time until the new torque limit is reached)

Change monitoring window

The monitoring window can be changed with the FXSW command. The changes take effect before the traversing motions of the block.

If a new monitoring window is programmed, not only the window width changes. If the axis has moved before the change, the reference point for the window center point also changes to the current actual position of the axis.

Function application for active continuous-path mode G64

The following machine data can be used to set that with the selection of the (FXS) function during active continuous-path mode (G64), no exact stop is triggered at the block change (G60):

MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgements for travel to fixed stop)

Bit	Value	Meaning
0	0	Start of the traversing motion without acknowledgement by the PLC
	1	Start of the traversing motion after acknowledgement by the PLC

Supplementary conditions

- The traversing motion to the fixed stop can be programmed as a path- or block-related or modal positioning axis motion.
- Travel to fixed stop can be be selected for several machine axes simultaneously.
- The travel path and the activation of the function must be programmed in one block in the part program.
- If "Travel to fixed stop" is activated via synchronized actions, the travel path and the activation of the function can be programmed in separate blocks.

Example

Requirement: Mapping of channel axis X on machine axis X1

Programming with channel axis identifier

```
Program code                    Comment
FXS[X] = 1                      ;   Selecting X ->X1
FXST[X] = 12.3                  ;   New torque X ->X1
FXSW[X] = 2000                  ;   New window for X -> X1
```

Programming with machine axis identifier

```
Program code                    Comment
FXS[X1] = 1                     ;   Selecting X1
FXST[X1] = 12.3                 ;   New torque for X1
FXSW[X1] = 2000                 ;   New window for X1
```

References:
Programming Manual, Fundamentals

5.2.2 Functional sequence

5.2.2.1 Selection

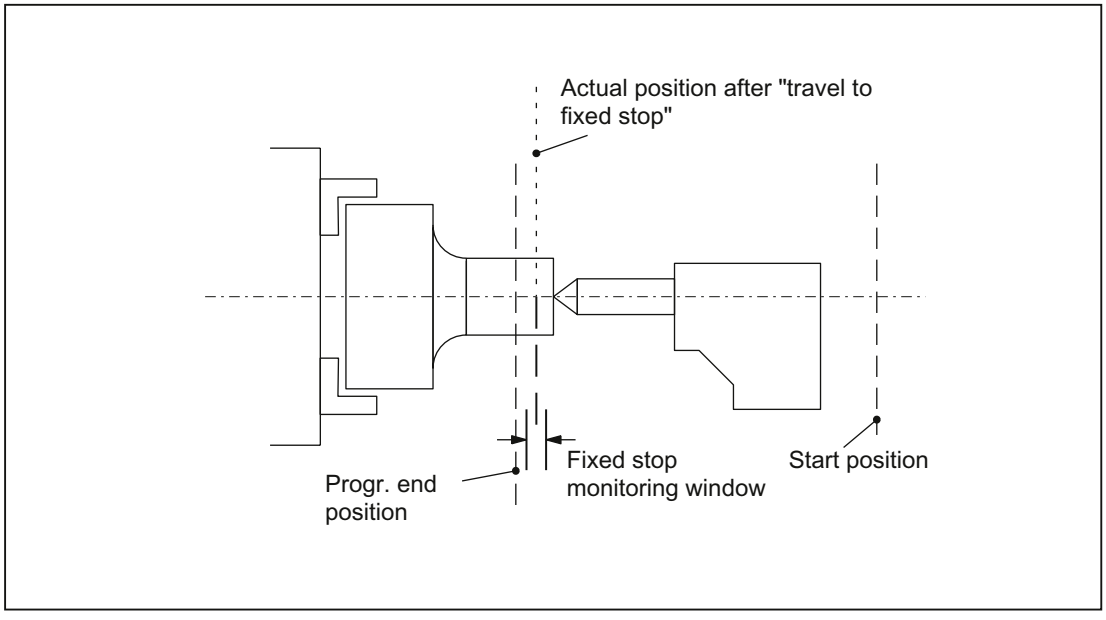


Figure 5-1 Example of travel to fixed stop

Procedure

The NC detects that the function "Travel to fixed stop" is selected via the command `FXS[x]=1` and signals the PLC using the IS DB31, ... DBX62.4 ("Activate travel to fixed stop") that the function has been selected.

If the machine data:

MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgements for travel to fixed stop)

is set correspondingly, the system waits for the acknowledgement of the PLC using the IS DB31, ... DBX3.1 ("Enable travel to fixed stop").

The programmed target position is then approached from the start position at the programmed velocity. The fixed stop must be located between the start and target positions of the axis/spindle. A programmed torque limit (clamping torque specified via `FXST[<axis>]`) is effective from the start of the block, i.e. the fixed stop is also approached with reduced torque. Allowance for this limitation is made in the NC through an automatic reduction in the acceleration rate.

If no torque has been programmed in the block or since the start of the program, then the value is valid in the axis-specific machine data:

MD37010 $MA_FIXED_STOP_TORQUE_DEF (default for fixed stop clamping torque)

is entered.

5.2.2.2 Fixed stop is reached

Procedure

When the axis has reached the fixed stop, the torque in the drive is increased up to the programmed clamping torque. How the control detects that the fixed stop has been reached, can be set via the following machine data:

MD37040 $MA_FIXED_STOP_BY_SENSOR = <value> (fixed stop detection via sensor)

<value>	Meaning
0	The fixed stop has been reached when the axial contour deviation (difference between actual and expected following error) has exceeded the set value in the machine data: MD37030 $MA_FIXED_STOP_THRESHOLD (threshold for fixed stop detection)
1	An external sensor detects when the fixed stop has been reached and informs the control via the following axial NC/PLC interface signal: DB31, ... DBX1.2 == 1 (sensor for fixed stop)
2	The fixed stop has been reached when the condition according to value == 0 or value == 1 applies.

Ineffective NC/PLC interface signals

When the axis is in the "Fixed stop reached" state, the following NC/PLC interface signals have no effect:

- DB31, ... DBX1.3 (axis/spindle disable)
- DB31, ... DBX2.1 (controller enable)

Internal processes

Once the NC has detected the "Fixed stop reached" state, it deletes the distance-to-go and the position setpoint is corrected. The controller enable remains active.

The PLC is then informed of the state using IS DB31, ... DBX62.5 ("Fixed stop reached").

If the machine data:

MD37060 $MA_FIXED_STOP_ACKN_MASK

is set correspondingly, then the system waits for the acknowledgement of the PLC using the IS DB31, ... DBX1.1 ("Acknowledge fixed stop reached").

The NC then executes a block change or considers the positioning motion to be completed, but still leaves a setpoint applied to allow the clamping torque to take effect.

The fixed stop monitoring function is activated as soon as the stop position is reached.

Monitoring window

If no fixed stop monitoring window was programmed in the block or from program start, then the value set in the machine data:

MD37020 $MA_FIXED_STOP_WINDOW_DEF

(default for fixed stop-monitoring window)

is entered.

If the axis leaves the position it was in when the fixed stop was detected, then alarm 20093 "Fixed stop monitoring has responded" is displayed and the "Travel to fixed stop" function deselected.

The window must be selected by the user such that the alarm is activated only when the axis leaves the fixed stop position.

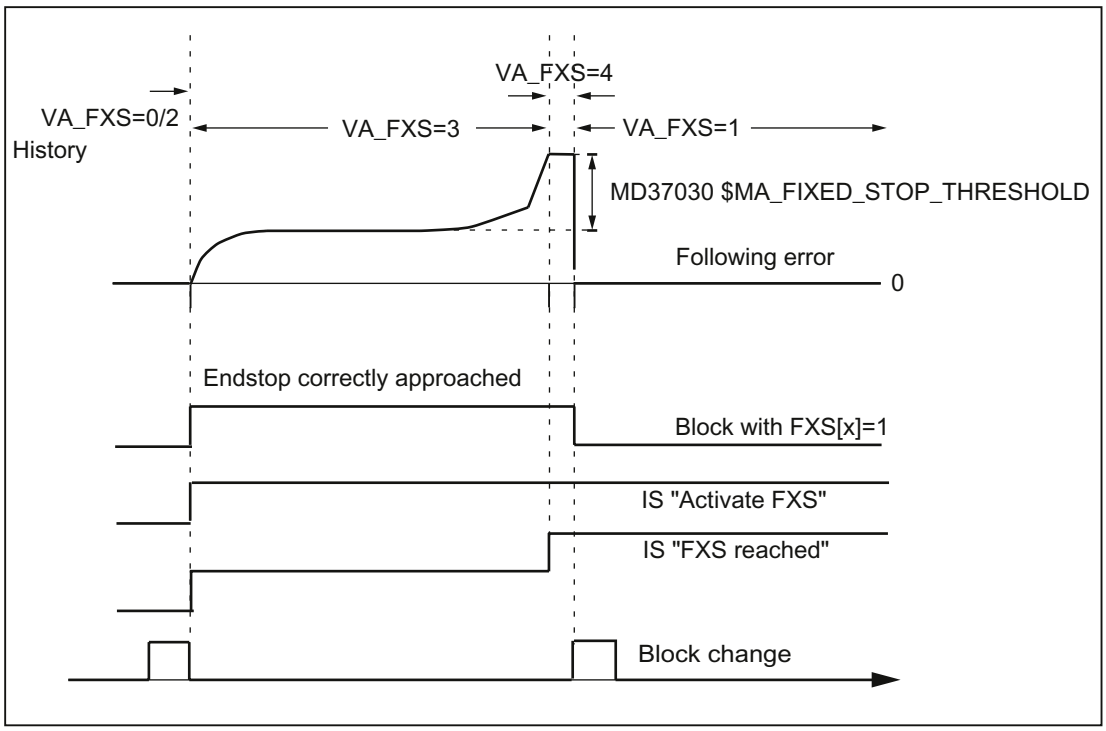


Figure 5-2 Fixed stop is reached

5.2.2.3 Fixed stop is not reached

Procedure

If the programmed end position is reached, without detecting the state "fixed stop reached", then depending on the state of the machine data:

MD37050 $MA_FIXED_STOP_ALARM_MASK = 0 (release of fixed stop alarms)

the alarm 20091 "Fixed stop not reached" is output.

Abort without alarm

The travel to fixed stop can be aborted by the PLC in the approach block without triggering an alarm (for example, when the operator actuates a key), if in the machine data:

MD37050 $MA_FIXED_STOP_ALARM_MASK = 2

the alarm 20094 is suppressed.

The Travel to fixed stop function is deselected in response to both "Fixed stop not reached" and "Fixed stop aborted".

Function abort

If the "Travel to fixed stop" function is aborted owing to a pulse disable, cancelation of PLC acknowledgements or a Reset in the approach block, the display or suppression of alarm 20094 can be controlled via machine data:

MD37050 $MA_FIXED_STOP_ALARM_MASK = 3

All other values ≤ 7 do not suppress any alarms.

Using the part program command `NEWCONF` a new setting can be activated.

Sequence in case of a fault or abnormal termination

Reset of the axial NC/PLC interface signal:

DB31, ... DBX62.4 = 0 (activate travel to fixed stop)

Depending on the setting in machine data:

MD37060 $MA_FIXED_STOP_ACKN_MASK

wait for the acknowledgement:

DB31, ... DBX3.1 == 0 (enable travel to fixed stop).

The torque limitation is then canceled and a block change executed.

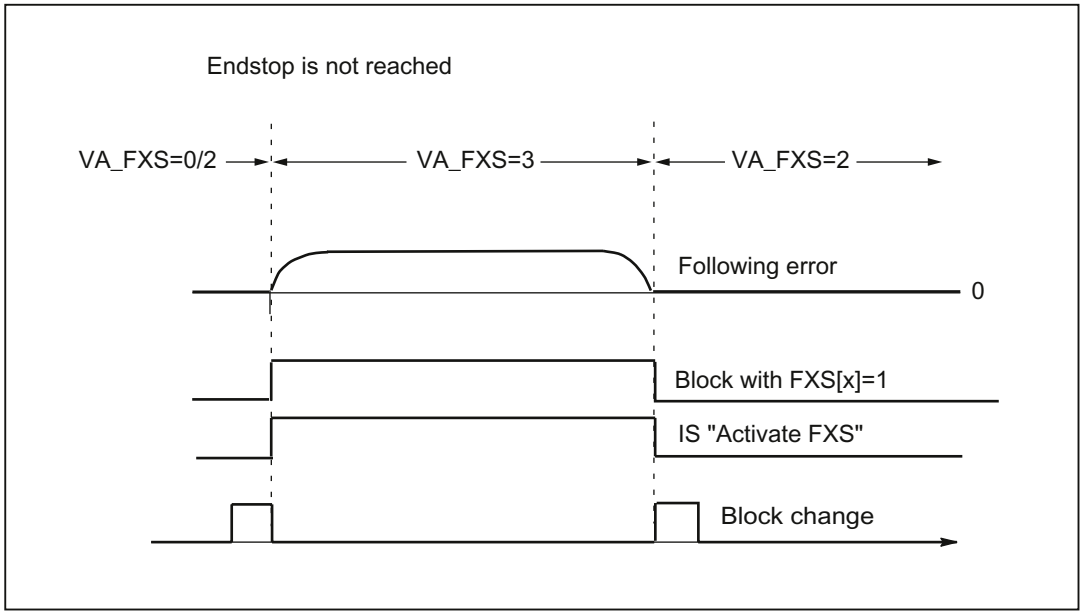


Figure 5-3 Fixed stop is not reached

5.2.2.4 Deselection

Procedure

With the deselection of the function FXS[<axis>]=0, a preprocessing stop (STOPRE) is triggered.

The torque limitation and monitoring of the fixed stop monitoring window are canceled.

The NC/PLC interface signals are reset:

DB31, ... DBX62.4 = 0 (activate travel to fixed stop)

DB31, ... DBX62.5 = 0 (fixed stop reached)

Depending on the machine data:

MD37060 $MA_FIXED_STOP_ACKN_MASK

wait for the acknowledgement:

DB31, ... DBX3.1 == 0 (enable travel to fixed stop)

DB31, ... DBX1.1 == 0 (acknowledge fixed stop reached)

The axis will then change to position control. The correction of the position setpoint is terminated and a synchronization to the new actual position is carried out. The axis can be traversed again.

Response when pulse enable is canceled

The pulse enable or pulse inhibit can be canceled via:

- Drive: Via terminal EP (enable pulses)
- NC/PLC interface signal: DB31, ... DBX21.7 ("pulse enable")

The behavior at the fixed stop can be set via the following machine data:

MD37002 $MA_FIXED_STOP_CONTROL (sequence control for travel to fixed stop)

Bit	Value	Meaning
0	Behavior for pulse disable at the stop	
	0	Travel to fixed stop is aborted
	1	Travel to fixed stop is interrupted, i.e. the drive is without power. Once the pulse disable is canceled again, the drive presses with the stop torque again. For control of the torque injection, see bit 1.
1	Behavior for pulse enable at the stop	
	0	The torque is injected suddenly.
	1	The torque is ramped up. See also: MD37012 $MA_FIXED_STOP_TORQUE_RAMP_TIME

If the pulse enable is deleted during an active deselection of the function (state: "Wait for PLC acknowledgements"), the torque limit is reduced to zero. During this phase, torque is no longer built up if the pulse enable is set again. Once the deselection has been completed, you can continue traversing as normally.

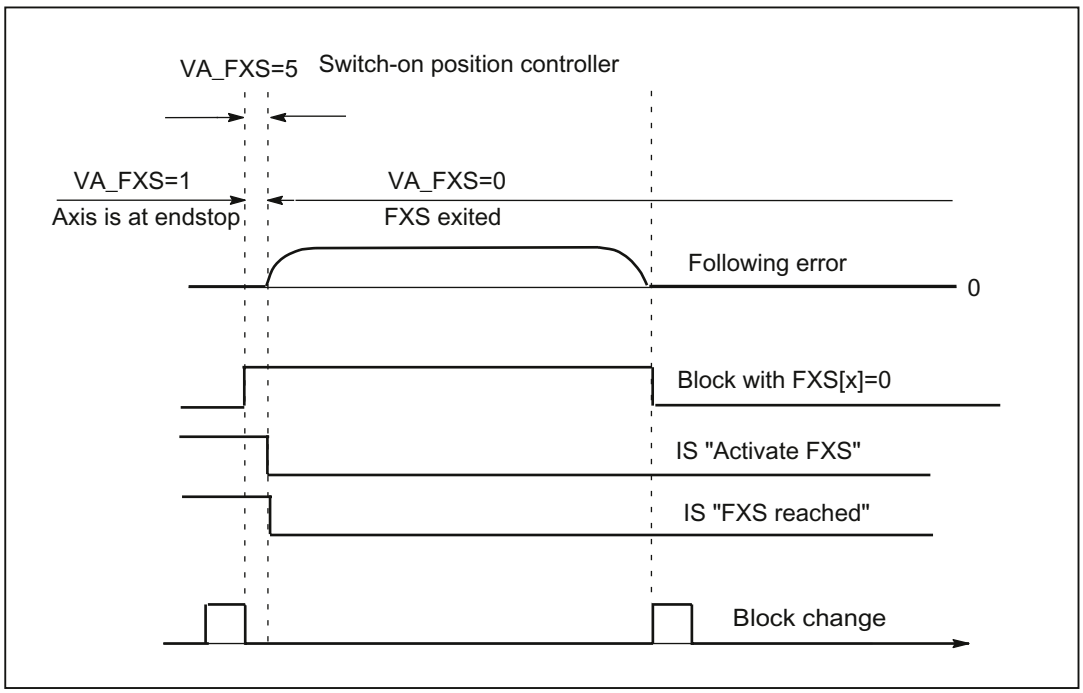


Figure 5-4 Fixed stop deselection

5.2.3 Behavior during block search

Block search with calculation

- If the target block is located in a program section in which the axis must stop at a fixed limit, then the fixed stop is approached if it has not yet been reached.
- If the target block is located in the program section in which the axis must not stop at a fixed limit, then the axis leaves the fixed stop if it is still positioned there.
- If the axis is in the "Fixed stop reached" state, message 10208 "Press NC start to continue the program" is displayed. The program can be continued with NC start.
- Clamping torque FXST and monitoring window FXSW both have the value that they have for normal program processing at the start of the target block.

Block search without calculation

The commands FXS, FXST and FXSW are ignored during the block search.

Effectiveness of FOCON/FOCOF

The state of the modal-acting torque/force reduction FOCON/FOCOF is maintained during the search and is effective in the approach block.

Block search with FXS or FOC

The user selects FXS or FOC in a program area of the target block in order to acquire all states and functions of the machining last valid. The NC starts the selected program in Program test mode automatically. After the target block has been found, the NC stops at the beginning of the target block, deselects Program test internally again and displays the Stop condition "Search target found" in its block display.

If FXS is located between the beginning of the program and the search target, the instruction is not executed by the NC. The motion is only simulated up to the programmed end point.

⚠ CAUTION
SERUPRO approach does not really take the FXS instruction into account. The approach to the programmed end position of the FXS block is only simulated **without torque limitation**.

The user can log the turning on and turning off of FXS in the part program. If necessary, the user can start an ASUB in order to activate or deactivate FXS in this SERUPRO-ASUP.

System variable

The state of the function can be read via the following system variable:

- $AA_FXS (desired state)
- $VA_FXS (actual state)

Value	Description
0	Axis not at fixed stop
1	Fixed stop successfully approached
2	Approach to fixed stop failed
3	Travel to fixed stop selection active
4	Fixed stop detected
5	Travel to fixed stop deselection active

SERUPRO $AA_FXS

During SERUPRO, $AA_FXS supplies the following values depending on the activation status of the "Travel to fixed stop" function:

FXS[<axis>]=<status>	
<status>	$AA_FXS
0 (switch off)	0 (axis not at fixed stop)
1 (switch on)	3 (travel to fixed stop selection active)

Outside of SERUPRO, both system variables always supply the same value.

Note

During SERUPRO, the system variable $AA_FXS only supplies the values 0 and 3. As a result, based on $AA_FXS, the program sequence can be changed with SERUPRO compared to the normal program execution for program branches.

SERUPRO $VA_FXS

During SERUPRO, the variable $VA_FXS always supplies the real state of axis on the machine.

SERUPRO ASUB Desired/actual state comparison

The current state of the "Travel to fixed stop" function can be determined in the SERUPRO ASUB via the system variables $AA_FXS and $VA_FXS, and the appropriate response initiated:

FXS_SERUPRO_ASUP.MPF

```
Program code
N100 WHEN ($AA_FXS[X]==3) AND ($VA_FXS[X]==0) DO FXS[X]=1
N200 WHEN ($AA_FXS[X]==0) AND ($VA_FXS[X]==1) DO FXS[X]=0
N1020 REPOSA
```

Displaying the REPOS offset

Once the search target has been found, the FXS state active on the machine is displayed for each axis via the following axial NC/PLC interface signals:

- DB31, ... DBX62.4 (activate travel to fixed stop)
- DB31, ... DBX62.5 (fixed stop reached)

Example:

If the axis is at the fixed stop and the target block is available after deselection of FXS, the new target position is displayed via DB31, ... DBX62.5 (fixed stop reached) as the REPOS offset.

REPOS and FXS

With REPOS, the functionality of FXS is repeated automatically and called FXS-REPOS in the following. This sequence is comparable to the FXS_SERUPRO_ASUP.MPF program. Whereby every axis is taken into account and the last torque programmed before the search target is used.

The user can treat FXS separately in a SERUPRO ASUB.

The following then applies:

Every FXS action executed in the SERUPRO ASUB automatically takes care of

$AA_FXS[<axis>] = $VA_FXS[<axis>].

This deactivates FXS-REPOS for axis X.

Deactivating FXS REPOS

FXS-REPOS is deactivated by:

- An FXS synchronized action which refers to REPOSA
- $AA_FXS[X] = $VA_FXS[X] in the SERUPRO_ASUB

Note

A SERUPRO ASUB **without** FXS treatment or no SERUPRO ASUB results automatically in FXS-REPOS.

CAUTION

FXS-REPOS traverse all path axes together to the target position. Axes with and without FXS treatment thus traverse together with the G code and feedrate valid in the target block. As a result, the fixed stop may be approached in rapid traverse(G0) or higher velocity.

FOC in the REPOS

FOC-REPOS behaves in the same way as FXS-REPOS.

A changing torque characteristic during the program preprocessing cannot be implemented with FOC-REPOS.

Example

Axis X is traversed from position 0 to 100. FOC is switched on every 20 millimeters for 10 millimeters. The resulting torque characteristic is generated with non-modal FOC and cannot be traced by FOC-REPOS. Axis X is traversed by FOC-REPOS with or without FOC in accordance with the last programming before the target block.

For programming examples of FXS "Travel to fixed stop", see Section "Program test (Page 511)".

5.2.4 Behavior for reset and function abort

NC reset

As long as the function is still not in the "Successful travel to fixed stop" state, the travel to fixed stop can be aborted with NC reset.

Even when the fixed stop has already has been approached, but the specified stop torque not yet fully reached, then the function can still be aborted with NC reset. The position setpoint of the axis is synchronized with the current actual position.

As soon as the function is in the "Successful travel to fixed stop" state, the function also remains active after the NC reset.

Function abort

A function abort can be triggered by the following events:

- Emergency stop

  Note

  NC and drive have no power during "Emergency stop", i.e. the PLC must react.

  ⚠ CAUTION

  It must be ensured that no dangerous machine situations occur while travel to fixed stop is active when an "Emergency stop" is triggered or reset.

  For example, behavior when setting and canceling the pulse enable:
  MD37002 $MA_FIXED_STOP_CONTROL, bit 0 (behavior for pulse disable at the stop)
  - Bit 0 = 0: Travel to fixed stop is aborted
  - Bit 0 = 1: Travel to fixed stop is interrupted, i.e. the drive is without power
    Once the pulse disable is canceled again, the drive presses with the stop torque again.

- Functional state: "Fixed stop not reached"
- Functional state: "Fixed stop aborted"
- Aborted by the PLC user program:
  DB31, ... DBX62.4 = 0 ("Activate travel to fixed stop")
- Cancellation of the pulse enable and machine data parameterization:
  MD37002 $MA_FIXED_STOP_CONTROL, bit 0 = 0 (see above)

5.2.5 Behavior with regard to other functions

Measurement with delete distance-to-go

"Travel to fixed stop" (FXS) cannot be programmed in a block with "Measurement with delete distance-to-go" (MEAS). Except when on function acts on a path axis while the other acts on a positioning axis or both functions act on positioning axes.

Contour monitoring

The axis contour monitoring function is inoperative while "Travel to fixed stop" is active.

Positioning axes

For "Travel to fixed stop" with positioning axes POSA, the block change is executed even when the positioning axis has not yet reached the fixed stop by this time.

Vertical axes

The "Travel to fixed stop" function can be used for vertical axes even when alarms are active.

If a function-specific alarm occurs for a vertical axis when traveling to fixed stop, the NC/PLC interface signal DB11, DBX6.3 (mode group ready) is not reset: This means that the corresponding drive is not de-energized.

This corresponds to an electronic weight compensation for the vertical axis and can be configured via the following machine data:

MD37052 $MA_FIXED_STOP_ALARM_REACTION

References

Further information on vertical axes can be found in:

- SINAMICS S120 Function Manual
- Function Manual, Extended Functions; Compensation (K3),
  Section: Electronic counterweight

5.2.6 Setting data

The values programmed via the function-specific FXS, FXST and FXSW commands are written block-synchronously to the following, immediately effective, axis-specific setting data:

Switching the function on/off

SD43500 $SA_FIXED_STOP_SWITCH (selection/deselection of travel to fixed stop)

Clamping torque

SD43510 $SA_FIXED_STOP_TORQUE (clamping torque)

Note

Clamping torque greater than 100%

A value for the clamping torque in SD43510 greater than 100% of the maximum motor torque is only advisable for a short time. In addition, the maximum motor torque is limited by the drive. For example, the following drive parameters have a limiting effect:

- p1520/p1521 upper torque limit/force limit / lower torque limit/force limit
- p1522/p1523 upper torque limit/force limit / lower torque limit/force limit
- p1530/p1531 power limit, motoring / power limit, regenerating
- p0640 current limit
- p0326 motor stall force correction factor

Detailed information on the drive parameters and the functions can be found in:

References

- SINAMICS S120/S150 List Manual
- SINAMICS S120 Function Manual

Monitoring window

SD43520 $SA_FIXED_STOP_WINDOW (monitoring window)

Default setting

The defaults for the setting data are set via the following machine data:

- Clamping torque:
  MD37010 $MA_FIXED_STOP_TORQUE_DEF (default clamping torque)
- Monitoring window:
  MD37020 $MA_FIXED_STOP_WINDOW_DEF (default monitoring window)

Effectiveness

The setting data for the clamping torque and monitoring window takes effect immediately. In this way, the clamping state can be adapted to the machining situation at any time by the operator or via the PLC user program.

References

Further detailed information on the machine and setting data can be found in:

List Manual, Detailed Machine Data Description

See also

Z1: NC/PLC interface signals (Page 1791)

5.2.7 System variable

Information on the status of the function can be read via the following system variables in part programs and synchronized actions:

- $AA_FXS (status, "Travel to fixed stop" desired state)
- $VA_FXS (status, "Travel to fixed stop" actual state)
- $VA_FXS_INFO (additional information for "Travel to fixed stop")
  The system variable contains additional information for the case that the travel to fixed stop has failed: $VA_FXS == 2

Note

Actual position at fixed stop

After reaching the fixed stop, the actual position of the machine axis can be read, for example, for evaluation or measuring purposes via the $AA_IM system variables.

Application example for $AA_FXS

In order that a block change is executed, no alarm should be triggered when a fault occurs. The cause can then be determined via the system variable and a specific reaction performed.

Requirement: MD37050 $MA_FIXED_STOP_ALARM_MASK = 0 (do not trigger an alarm)

Program code

```
X300 Y500 F200 FXS[X1]=1 FXST[X1]=25 FXSW[X1]=5
; IF fixed stop == reached => normal case
IF $AA_FXS[X1]==2 GOTOF FXS_ERROR
 ; Normal case
 G01 X400 Y200
 ...
 GOTOF PROG_END
; ELSE error handling
 FXS_ERROR: ...
PROG_END: M30
```

5.2.8 Fixed stop alarms

Alarm 20091 "Fixed stop not reached"

If the fixed stop position is not reached during travel to fixed stop, alarm 20091 "Fixed stop not reached" is displayed and a block change executed.

Alarm 20092 "Travel to fixed stop is still active"

If there is a travel request or renewed function selection for the axis after the fixed stop has been reached, alarm 20092 "Travel to fixed stop is still active" is displayed.

Alarm 20093 "Standstill monitoring at fixed stop has triggered"

If an axis has reached the fixed stop and is then moved out of this position by more than the value specified in the setting data

SD43520 FIXED_STOP_WINDOW (fixed stop monitoring window)

alarm 20093 "Standstill monitoring at fixed stop has triggered" is displayed, travel to fixed stop for this axis is deselected and the following system variable set:
$AA_FXS[x] = 2

Alarm 20094 "Function has been aborted"

The travel to fixed stop is aborted if the clamping torque can no longer be applied due to the cancellation of the pulse enable, or the requested acknowledgement signal at the NC/PLC interface has been reset:

- Acknowledgement signal required: MD37060 $MA_FIXED _STOP_ACKN_MASK, bit 0 = 1
- Acknowledgement signal: DB31, ... DBX3.1 == 0 (enable travel to fixed stop)

Enabling the fixed stop alarms

The following machine data can be use to set whether the fixed stop alarms

- Alarm 20091 "Fixed stop not reached",
- Alarm 20094 "Fixed stop aborted"

are displayed:

MD37050 $MA_FIXED_STOP_ALARM_MASK (enable of the fixed stop alarms)

Settable functional behavior for fixed stop alarms

The following machine data can be used to set that the function is not aborted when function-specific alarms occur:

- Alarm 20090 Travel to fixed stop not possible
- Alarm 20091 Fixed stop not reached
- Alarm 20092 Travel to fixed stop is still active
- Alarm 20093 Standstill monitoring at fixed stop has triggered
- Alarm 20094 Travel to fixed stop aborted

MD37052 $MA_FIXED_STOP_ALARM_REACTION (reaction to fixed stop alarms)

Alarm suppression after new programming

Travel to fixed stop can be used for simple measuring processes.

For example, it is possible to carry out a check for tool breakage by measuring the tool length by traversing onto a defined obstacle. To do so, the fixed stop alarm must be suppressed. When the function for clamping workpieces is then used "normally," the alarm can be activated using part program commands.

5.2.9 Travel with limited torque/force FOC

Function

For applications in which torque or force are to be changed dynamically depending on the travel or on the time or on other parameters (e.g. pressing), the functionality FOC (Force Control) is provided.

Force/travel or force/time profiles are thus possible in the resolution for the interpolation cycle.

The function allows torque/force to be modified at any time using synchronized actions.

The function can be activated modally or non-modally.

Availability

System	Availability
SINUMERIK 840D sl	Standard (basic scope)
SINUMERIK 828D	Option

Modal activation (FOCON/FOCOF)

The activation of the function after POWER ON and RESET is determined by the machine data:

MD37080 $MA_FOC_ACTIVATION_MODE (controlling the initial setting of the modal limitation of torque/force)

Bit	Meaning	
0	Effective after POWER ON:	
	= 0	FOCOF
	= 1	FOCON
1	Effective after RESET:	
	= 0	FOCOF
	= 1	FOCON

FOCON: Activation of the modally effective torque/force limitation

FOCOF: Disables the torque/force limitation

The modal activation acts beyond the program end.

If already programmed, the torque/force set with FXST is effective.

FXST can be programmed irrespectively of FOCON; it comes into effect, however, only after the function has been activated.

Programming

The programming of the axis is carried out in square brackets.

The following are permissible:

- Geometry axis identifiers
- Channel axis identifiers
- Machine axis identifiers

Example:

```
Program code                    Comment
N10 FOCON[X]                    ; Modal activation of the torque limit
N20 X100 Y200 FXST[X]=15        ; X travels with reduced torque (15%)
N30 FXST[X]=75 X20              ; Changing the torque to 75%.
                                ; X travels with this reduced torque.
N40 FOCOF[X]                    ; Disable torque limit
```

Block-related limit (FOC)

The part program command FOC activates the torque limit for a block.

An activation from a synchronized action takes effect up to the end of the current part program block.

Priority FXS/FOC

An activation of FXS with FOC active has priority, i.e. FXS is executed.

A deselection of FXS will cancel the clamping.

A modal torque/force limitation remains active.

After POWER ON the activation takes effect with the machine data:

MD37010 $MA_FIXED_STOP_TORQUE_DEF

This torque can be changed at any time by programming FXST.

Synchronized actions

The language commands FOC, FOCON, FOCOF can also be programmed in synchronized actions as the commands for "Travel to fixed stop".

Determine FOC status

The activation status can be read at any time via the status variable $AA_FOC.

If FXS is also activated, the status is not changed.

Value	Meaning
0	FOC not active
1	FOC modal active
2	FOC non-modal active

Determine torque limit status

The system variables $VA_TORQUE_AT_LIMIT can be used at any time to read in systems whether the currently active torque corresponds to the specified torque limit.

Value	Meaning
0	Effective torque < torque limit value
1	Effective torque has reached the torque limit value

Restrictions

The FOC function is subject to the following restrictions:

- The change of the torque/force limitation represented as an acceleration limitation is taken into account in the traversing motion **only at the block limits** (see command ACC).
- Only FOC:

  No monitoring is possible from the NC/PLC interface to check that the active torque limit has been reached.
- If the acceleration limitation is not adapted accordingly, an increase in the following error occurs during the traversing motion.
- If the acceleration limitation is not adapted accordingly, the end-of-block position is possibly reached later than specified in:
  MD36040 $MA_STANDSTILL_DELAY_TIME
  .

  The machine data:
  MD36042 $MA_FOC_STANDSTILL_DELAY_TIME
  is introduced for this and monitored in this status.

Possible application for link and container axes

All axes that can be traversed in a channel, i.e. also link axes and container axes, can be traversed to fixed stop.

References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

The status of the machine axis is kept in the case of a container rotation, i.e. a clamped machine axis remains at the stop.

If a modal torque limitation has been activated with FOCON, this is kept for the machine axis even after a container rotation.

5.3 Examples

Static synchronized actions

Travel to fixed stop (FXS), initiated by a synchronized action.

```
Program code                        Comment
N10 IDS=1 WHENEVER                  ; Activate static synchronized action:
(($R1==1) AND                         By the setting of $R1=1
($AA_FXS[Y]==0)) DO                   for
$R1=0 FXS[Y]=1                        the axis Y FXS is activated
FXST[Y]=10                            the active torque is reduced to 10%
FA[Y]=200                             and a travel motion to
POS[Y]=150                            direction of the stop started

N11 IDS=2 WHENEVER                  ; Once the stop has been recognized
($AA_FXS[Y]==4) DO                    ($AA_FXS[Y]==4), the torque is
FXST[Y]=30                            increased to 30%

N12 IDS=3 WHENEVER                  ; after reaching the stop
($AA_FXS[Y]==1) DO                    the torque becomes dependent
FXST[Y]=$R0                           on R0 controlled

N13 IDS=4 WHENEVER                  ; Deselection depending on
(($R3==1) AND                         on R3 and
($AA_FXS[Y]==1)) DO                   reverse
FXS[Y]=0
FA[Y]=1000 POS[Y]=0

N20 FXS[Y]= 0                       ; Normal program run
G0 G90 X0 Y0

N30 RELEASE(Y)                      ; Enable axis Y for the motion in
                                      the synchronized action

N40 G1 F1000 X100                   ; Motion of another axis
N50 ......

N60 GET(Y)                          ; Include axis Y again in the
                                      path group
```

Multiple selection

A selection may only be carried out once. If the function is called once more due to faulty programming (`FXS[Axis]=1`) the alarm 20092 "Travel to fixed stop still active" is initiated.

Programming code that scans $AA_FXS[] or a separate marker (here R1) in the condition will ensure that the function is not activated more than once.

Programming example (part program fragment):

```
Program code
N10 R1=0
N20 IDS=1 WHENEVER ($R1 == 0 AND $AA_IW[AX3]>7) DO R1=1 FXS[AX1]=1
FXST[AX1] = 12
```

Block-related synchronized actions

By programming a block-related synchronized action, travel to fixed stop can be connected during an approach motion.

Programming example:

```
Program code                                Comment
N10 G0 G90 X0 Y0
N20 WHEN $AA_IW[X]>17 DO FXS[X]=1           ; If X reaches a position greater
N30 G1 F200 X100 Y110                       ; 17 mm, FXS is activated
```

5.4 Data lists

5.4.1 Machine data

5.4.1.1 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
36042	FOC_STANDSTILL_DELAY_TIME	Delay time, standstill monitoring for `active torque/force limiting`
37000	FIXED_STOP_MODE	Travel to fixed stop mode
37002	FIXED_STOP_CONTROL	Sequence monitoring for travel to fixed stop
37010	FIXED_STOP_TORQUE_DEF	Fixed stop clamping torque default setting
37012	FIXED_STOP_TORQUE_RAMP_TIME	Time until the modified torque limit is reached
37020	FIXED_STOP_WINDOW_DEF	Default for fixed stop monitoring window
37030	FIXED_STOP_THRESHOLD	Threshold for fixed stop detection
37040	FIXED_STOP_BY_SENSOR	Fixed stop detection via sensor
37050	FIXED_STOP_ALARM_MASK	Enabling the fixed stop alarms
37052	FIXED_STOP_ALARM_REACTION	Reaction to fixed stop alarms
37060	FIXED_STOP_ACKN_MASK	Monitoring of PLC acknowledgments for travel to fixed stop
37070	FIXED_STOP_ANA_TORQUE	Torque limit on fixed stop approach for analog drives
37080	FOC_ACTIVATION_MODE.	Initial setting of the modal torque/force limiting

5.4.2 Setting data

5.4.2.1 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43500	FIXED_STOP_SWITCH	Selection of travel to fixed stop
43510	FIXED_STOP_WINDOW	Fixed stop clamping torque
43520	FIXED_STOP_TORQUE	Fixed stop monitoring window

5.4.3 Signals

5.4.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Acknowledge fixed stop reached	DB31, ...DBX1.1	DB380x.DBX1.1
Sensor for fixed stop	DB31, ...DBX1.2	DB380x.DBX1.2
Axis/spindle disable	DB31, ...DBX1.3	DB380x.DBX1.3
Controller enable	DB31, … .DBX2.1	DB380x.DBX2.1
Enable travel to fixed stop	DB31, … .DBX3.1	DB380x.DBX3.1

5.4.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate travel to fixed stop	DB31, ...DBX62.4	DB390x.DBX2.4
Fixed stop reached	DB31, ...DBX62.5	DB390x.DBX2.5

G2: Velocities, setpoint / actual value systems, closed-loop control

6

6.1 Brief description

The description of functions explains how to parameterize a machine axis in relation to:

- Actual-value/measuring systems
- Setpoint system
- Operating accuracy
- Travel ranges
- Axis velocities
- Control parameters

6.2 Velocities, traversing ranges, accuracies

6.2.1 Velocities

Maximum path and axis velocities and spindle speed

The maximum path and axis velocities and spindle speed are influenced by the machine design, the dynamic response of the drive and the limit frequency of the actual-value acquisition (encoder limit frequency).

The maximum axis velocity is defined in machine data:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

The maximum permissible spindle speed is specified using machine data:

MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

For explanations, see Section "S1: Spindles (Page 1383)".

With a high feedrate (resulting from programmed feedrates and feedrate override), the maximum path velocity is limited to V_{max}.

This automatic feedrate limiting can lead to a drop in velocity over several blocks with programs generated by CAD systems with extremely short blocks.

Example:

Interpolation cycle = 12 ms

```
N10 G0 X0 Y0; [mm]

N20 G0 X100 Y100; [mm]
```

⇒ Path length programmed in block = 141.42 mm

⇒ V_{max} = (141.42 mm/12 ms) * 0.9 = 10606.6 mm/s = 636.39 m/min

Minimum path, axis velocity

The following restriction applies to the minimum path or axis velocity:

$$V_{min} \geq \frac{10^{-3}}{\text{Computational resolution } [\frac{\text{Incr.}}{\text{mm or degrees}}] * \text{interpolation cycle [s]}}$$

The computational resolution is defined using machine data:
MD10200 $MN_INT_INCR_PER_MM (computational resolution for linear positions)

or
MD10210 $MN_INT_INCR_PER_DEG (computational resolution for angular positions)

If V_{min} is not reached, no traversing is carried out.

Example:

MD10200 $MN_INT_INCR_PER_MM = 1000 [incr. /mm];

Interpolation cycle = 12 ms;

$\Rightarrow V_{min} = 10^{-3}/(1000\ ^{incr}/_{mm} \times 12\ ms) = 0.005\ ^{mm}/_{min}$;

The value range of the feedrates depends on the computational resolution selected.

For the standard assignment of machine data:

MD10200 $MN_INT_INCR_PER_MM
(computational resolution for linear positions) (1000 incr./mm)

or

MD10210 $MN_INT_INCR_PER_DEG

(computational resolution for angular positions) (1000 incr./deg.)

The following value range can be programmed with the specified resolution:

Range of values for path feed F and geometry axes:	
Metric system:	Inch system:
0.001 ≤ F ≤ 999,999.999 [mm/min, mm/rev, degrees/min, degrees/rev]	0.001 ≤ F ≤ 399,999.999 [inch/min, inch/rev]

Range of values for feedrate for positioning axes:	
Metric system:	Inch system:
0.001 ≤ FA ≤ 999,999.999 [mm/min, mm/rev, degrees/min, degrees/rev]	0.001 ≤ FA ≤ 399,999.999 [inch/min, inch/rev]

Range of values for spindle speed S:
0.001 ≤ S ≤ 999,999.999 [rpm]

If the computational resolution is increased/decreased by a factor, then the value ranges change accordingly.

6.2.2 Traversing ranges

Range of values of the traversing ranges

The range of values of the traversing range depends on the computational resolution selected.

For the standard assignment of machine data:

MD10200 $MN_INT_INCR_PER_MM

(computational resolution for linear positions) (1000 incr./mm)

or

MD10210 $MN_INT_INCR_PER_DEG

(computational resolution for angular positions) (1000 incr./deg.)

The following value range can be programmed with the specified resolution:

Table 6- 1 Traversing ranges of axes

	G71 [mm, degrees]	G70 [inch, degrees]
	Range	Range
Linear axes X, Y, Z, etc.	∓ 999,999.999	∓ 399,999.999
Rotary axes A, B, C, etc.	∓ 999,999.999	∓ 999,999.999
Interpolation parameters I, J, K	∓ 999,999.999	∓ 399,999.999

The unit of measurement of rotary axes is always degrees.

If the computational resolution is increased/decreased by a factor of 10, the ranges of values change accordingly.

The traversing range can be limited by software limit switches and working areas (see Section "A3: Axis Monitoring, Protection Zones (Page 85)").

For special features for a large traversing range for linear and rotary axes, see Section "R1: Referencing (Page 1319)".

The traversing range for rotary axes can be limited via machine data.

References:

Function Manual, Extended Functions; Rotary Axes (R2)

6.2.3 Positioning accuracy of the control system

Actual-value resolution and computational resolution

The positioning accuracy of the control depends on the actual-value resolution (=encoder increments/(mm or degrees)) and the computational resolution (=internal increments/(mm or degrees)).

The coarse resolution of these two values determines the positioning accuracy of the control.

The input resolution, interpolator and position-control cycle selections have no effect on this accuracy.

As well as limiting using MD32000, the control limits the maximum path velocity in relation to the situation and according to the following formula:

$$\frac{\text{Internal increments / mm}}{\text{Encoder increments / mm}} = \frac{1}{\text{ENC\_RESOL [n]} * \text{ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \frac{\text{DRIVE\_AX\_RATIO\_DENOM [n]}}{\text{DRIVE\_AX\_RATIO\_NUMERA [n]}} * \text{LEADSCREW\_PITCH} * \text{INT\_INCR\_PER\_MM}$$

6.2.4 Input/display resolution, computational resolution

Resolutions: Differences

Resolutions, e.g. resolutions of linear and angular positions, velocities, accelerations and jerk, must be differentiated as follows:

- Input resolution

  Data is input via the control panel or part programs.

- Display resolution

  Data is displayed via the control panel.

- Computational resolution

  Data input via the control panel or part program is displayed internally.

The input and display resolution is determined by the specified operator panel front used, whereby the display resolution for position values with the machine data:

MD9004 $MM_DISPLAY_RESOLUTION (display resolution)

can be changed.

The machine data:

MD9011 $MM_DISPLAY_RESOLUTION_INCH (display resolution for INCH measuring system)

can be used to configure the display resolution for position values with inch setting.

This allows you to display up to six decimal places with the inch setting.

For the programming of part programs, the input resolutions listed in the Programming Guide apply.

The desired computational resolution is defined using the machine data:

MD10200 $MN_INT_INCR_PER_MM (computational resolution for linear positions)

and

MD10210 $MN_INT_INCR_PER_ DEG (computational resolution for angular positions).

It is independent of the input/display resolution but should have at least the same resolution.

The maximum number of places after the decimal point for position values, velocities, etc., in the part program and the number of places after the decimal point for tool offsets, zero offsets, etc. (and therefore also for the maximum possible accuracy) is defined by the computational resolution.

The accuracy of angle and linear positions is limited to the computational resolution by rounding the product of the programmed value with the computational resolution to an integer number.

To make the rounding clear, powers of 10 should be used for the calculation resolution.

Example of rounding:

Computational resolution: 1000 incr./mm

Programmed path: 97.3786 mm

Effective value: 97.379 mm

Example of programming in the $^1/_{10}$ µm range:

All the linear axes of a machine are to be programmed and traversed within the range of values 0.1 to 1000 µm.

⇒ In order to position accurately to 0.1 µm, the computational resolution must be set to ≥ 10^4 incr./mm.

⇒ MD10200 $MN_INT_INCR_PER_MM = 10000 [incr./mm]:

⇒ Example of related part program:

Program code	**Comment**
`N20 G0 X 1.0000 Y 1.0000`	`; Axes travel to the position X=1.0000 mm, Y=1.0000 mm;`
`N25 G0 X 5.0002 Y 2.0003`	`; Axes travel to the position X=5.0002 mm, Y=2.0003 mm`

6.2.5 Scaling of physical quantities of machine and setting data

Input/output units

Machine and setting data that possess a physical quantity are interpreted in the input/output units below depending on whether the metric or inch system is selected:

Physical quantity:	Input/output units for standard basic system:	
	Metric	Inch
Linear position	1 mm	1 inch
Angular position	1 degree	1 degree
Linear velocity	1 mm/min	1 inch/min
Angular velocity	1 rpm	1 rpm
Linear acceleration	1 m/s^2	1 $inch/s^2$
Angular acceleration	1 $rev./^{s2}$	1 $rev./s^2$
Linear jerk	1 m/s^3	1 $inch/s^3$
Angular jerk	1 $rev./s^3$	1 $rev./s^3$
Time	1 s	1 s
Position controller servo gain	1/s	1/s
Rev. feedrate	1 mm/rev	inch/rev
Compensation value linear position	1 mm	1 inch
Compensation value angular position	1 degree	1 degree

The units listed below are used for storage. The control always uses these units internally irrespective of the basic system selected.

Physical quantity:	Internal unit:
Linear position	1 mm
Angular position	1 degree
Linear velocity	1 mm/s
Angular velocity	1 deg./s
Linear acceleration	1 mm/s^2
Angular acceleration	1 $degree/s^2$
Linear jerk	1 mm/s^3
Angular jerk	1 $degree/s^3$
Time	1 s
Position controller servo gain	1/s
Rev. feedrate	1 mm/degree
Compensation value linear position	1 mm
Compensation value angular position	1 degree

The user can define different input/output units for machine and setting data.

For this, the machine data:

MD10220 $MN_SCALING_USER_DEF_MASK

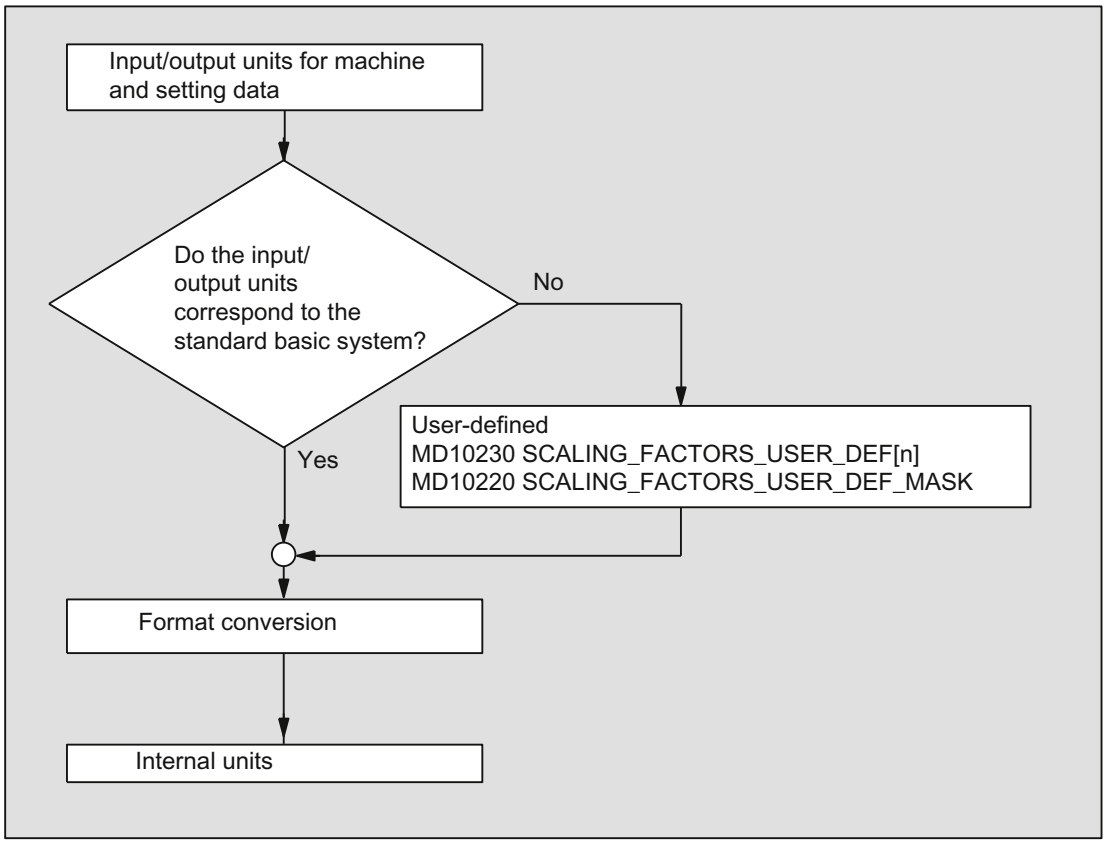
(activation of scaling factors)

and

MD10230 $MN_SCALING_FACTORS_USER_DEF[n]

(Scaling factors of physical quantities)

allow you to set the adaptation between the newly selected input/output units and the internal units.



The following applies:

Selected input/output unit = MD10230 * internal unit

In the machine data:

MD10230 $MN_SCALING_FACTORS_USER_DEF[n]

the selected input/output unit printed in each case in the internal units 1mm, 1 degree and 1 s must be input.

Example 1:

Machine data input/output of the linear velocities is to be in m/min instead of mm/min (initial state).

(The internal unit is mm/s)

⇒ The scaling factor for the linear velocities is to differ from the standard setting. For this in the machine data:

MD10220 $MN_SCALING_USER_DEF_MASK

the bit number 2 must be set.

⇒ MD10220 $MN_SCALING_USER_DEF_MASK = 'H4'; (bit no. 2 as hex value)

⇒ The scaling factor for the linear velocities is to differ from the standard setting. For this in the machine data:

MD10220 $MN_SCALING_USER_DEF_MASK

the bit number 2 must be set.

⇒ MD10220 $MN_SCALING_USER_DEF_MASK = 'H4'; (bit no. 2 as hex value)

⇒ The scaling factor is calculated using the following formula:

$$\text{MD10230 SCALING\_FACTORS\_USER\_DEF[n]} = \frac{\text{Input/output unit selected}}{\text{Internal unit}}$$

$$\text{MD10230 SCALING\_FACTORS\_USER\_DEF[n]} = \frac{1\,\frac{\text{m}}{\text{min}}}{1\,\frac{\text{mm}}{\text{s}}} = \frac{\frac{1000\ \text{mm}}{60\ \text{s}}}{1\,\frac{\text{mm}}{\text{s}}} = \frac{1000}{60} = 16.667;$$

$$\Rightarrow \text{MD10230 SCALING\_FACTORS\_USER\_DEF[n]} = 16.667$$

Index n defines the "linear velocity" in the "Scaling factors of physical quantities" list.

Example 2:

In addition to the change in Example 1, the machine data input/output of linear accelerations must be in ft/s² instead of m/s² (initial state).
(The internal unit is mm/s².)

⇒ MD10220 SCALING_USER_DEF_MASK = H14 ; (Bit No. 4 and Bit No. 2 from example 1 as hex value)

⇒ MD10220 SCALING_FACTORS_USER_DEF[4] $= \frac{1\,\frac{ft}{s^2}}{1\,\frac{mm}{s^2}} = \frac{12*25{,}4\,\frac{mm}{s^2}}{1\,\frac{mm}{s^2}} = 304{,}8$;

⇒ MD10230 SCALING_FACTORS_USER_DEF[4] = 304,8

Index 4 defines the "linear acceleration" in the "Scaling factors of physical quantities" list.

6.3 Metric/inch measuring system

6.3.1 Conversion of basic system by part program

Programmable switchover in the measuring system

The basic system can be switched over within a part program via the G functions G70/G71/G700/G710 (G group 13). The programmed measuring system (G70/G71/G700/G710) and the basic system may be identical or different at any time. When the measuring system is switched over within a particular section of the part program, this would enable an inch thread to be processed on a workpiece within a metric basic system, for example.

The following section of the part program is executed in "metric" in the basic system:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC = 1

```
Program code          Comment
N100 G71              ;   Switchover to metric processing
                      ;   The conversion factor does not come into effect because the
                      ;   programmed
                          measuring system does not differ from the basic system
....                  ;   Metric processing
N200 G70              ;   Switchover to processing in inches
                      ;   The conversion factor comes into effect
...                   ;   Processing in inches
N300                  ;   Switchover to metric processing
...                   ;   Metric processing
```

Initial state of the G functions

The initial state for the G functions can be set via the following machine data on a channel-specific basis:
MD20150 $MC_GCODE_RESET_VALUES[**12**] (reset position for G group **13**)

When changing the measuring system via the HMI user interface, the reset position is automatically preconfigured for compatibility with the new measuring system via G700 or G710.

Displaying length-related data on the HMI

Length-related data is displayed on the HMI in the configured basic system MD10240 $MN_SCALING_SYSTEM_IS_METRIC (metric basic system).

The following length-related data is displayed in the configured basic system:

- Machine data
- Data in the machine coordinate system
- Tool data
- Zero offsets

The following length-related data is displayed in the programmed measuring system:

- Data in the workpiece coordinate system

Reading in part programs from external sources

If part programs, including data sets (zero offsets, tool offsets, etc.), programmed in a different measuring system from the basic system are read in from an external source, the initial state must first be changed via machine data MD10240.

NC/PLC interface signals

In the case of NC/PLC interface signals containing dimension information, e.g. feedrate for path and positioning axes, data exchange is carried out with the PLC in the configured basic system.

G functions G700/G710

The G functions G700/G710 extend the functionality of G70/G71 as follows:

1. The feedrate is interpreted in the programmed measuring system:
   - G700: length parameters [inch]; feedrates [inch/min]
   - G710: length parameters [mm]; feedrates [mm/min]

   The programmed feedrate is modal and therefore remains active after subsequent G70/G71/G700/G710 commands. If the feedrate is to apply in the new G70/G71/G700/G710 context, it must be re-programmed.
2. System variables and machine data specifying lengths in the part program are read and written in the programmed measuring system.

Differences during the reading and writing of machine data and system variables

The following differences exist between G70/G71 and G700/G710 in terms of reading and writing machine data and system variables in the part program:

- G70/G71: Reading and writing takes place in the configured basic system.
- G700/G710: Reading and writing takes place in the configured measuring system.

Example
The following part program is executed with an initial metric state:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC = 1

```
Program code                                    Comment
N100 R1=0 R2=0                                  ;
N120 G01 G70 X1 F1000                           ;   Prog. meas. system: inch
N130 $MA_LUBRICATION_DIST[X]=10                 ;   MD = 10 [mm] (basic system)
N150 IF ($AA_IW[X]>$MA_LUBRICATION_DIST[X])     ;   SYS [mm] > MD [mm] (both basic system)
N160 R1=1                                       ;
N170 ENDIF                                      ;
N180 IF ($AA_IW[X]>10)                          ;   SYS [mm] (basic system) > 10 [inch]
                                                ;   (prog. meas. system)
N190 R2=1                                       ;
N200 ENDIF                                      ;
N210 IF ( (R1+R2) = 1 )                         ;   Alarm if only one of the two
                                                ;   conditions (N150, N180) is TRUE
N220 SETAL(61000)                               ;
N230 ENDIF                                      ;
N240 M30                                        ;
```

N120: if G70 is replaced by G700, alarm 61000 (N220) does not occur.

Synchronized actions

To ensure in the case of synchronized actions that the current part program context does not determine the measuring system used in the condition and/or action part, the measuring system must be defined within the synchronized action (condition and/or action parts). This is the only way of achieving defined, reproducible behavior in the use of length-related data within a synchronized action.

Example 1

The measuring system is not specified within the synchronized action. Therefore, the traversing motion of the X axis takes place in the measuring system of the configured initial state:

```
Program code                                    Comment
N100 R1=0                                       ;
N110 G0 X0 Z0                                   ;
N120 WAITP(X)                                   ;
N130 ID=1 WHENEVER $R1==1 DO POS[X]=10          ;   X = 10 inch or mm, depending on the
                                                ;   rest of the part program
N140 R1=1                                       ;
N150 G71 Z10 F10                                ;   Z = 10 mm         X = 10 mm
N160 G70 Z10 F10                                ;   Z = 10 inch       X = 10 inch
N170 G71 Z10 F10                                ;   Z = 10 mm         X = 10 mm
N180 M30                                        ;
```

Example 2

The "metric" measuring system is explicitly programmed with `G71` within the synchronized action. This means the traversing motion of the X axis takes place in the metric measuring system:

```
Program code                                    Comment
N100 R1=0                                       ;
N110 G0 X0 Z0                                   ;
N120 WAITP(X)                                   ;
N130 ID=1 WHENEVER $R1==1 DO G71 POS[X]=10      ;   X = 10 mm, independent of the
                                                ;   rest of the part program
N140 R1=1                                       ;
N150 G71 Z10 F10                                ;   Z = 10 mm       X = 10 mm
N160 G70 Z10 F10                                ;   Z = 10 inch     X = 10 mm
N170 G71 Z10 F10                                ;   Z = 10 mm       X = 10 mm
N180 M30                                        ;
```

Reading and writing of data in the case of G70/G71 and G700/G710 in the part program

Data area	G70/G71		G700/G710	
	Read	Write	Read	Write
Display, decimal places (WCS)	P	P	P	P
Display, decimal places (MCS)	G	G	G	G
Feedrates	G	G	P	P
Positional data X, Y, Z	P	P	P	P
Interpolation parameters I, J, K	P	P	P	P
Circle radius (CR)	P	P	P	P
Polar radius (RP)	P	P	P	P
Thread pitch	P	P	P	P
Programmable FRAME	P	P	P	P
Settable FRAMES	G	G	P	P
Basic frames	G	G	P	P
External zero offsets	G	G	P	P
Axial preset offset	G	G	P	P
Operating range limit (G25/G26)	G	G	P	P
Protection zones	P	P	P	P
Tool offsets	G	G	P	P
Length-related machine data	G	G	P	P
Length-related setting data	G	G	P	P
Length-related system variables	G	G	P	P
GUD	G	G	G	G
LUD	G	G	G	G
PUD	G	G	G	G
R parameters	G	G	G	G
Siemens cycles	P	P	P	P
Jog/handwheel increment factor	G	G	G	G
P: Data is read/written in the programmed measuring system G: Writing/reading takes place in the configured basic system.				

NOTICE

Read position data in synchronized actions

If a measuring system has not been explicitly programmed in the synchronized action (condition component and/or action component) **length-related position data** in the synchronized action are always read in the **parameterized basic system**.

References:
Programming Manual, Fundamentals; List of Addresses

NC-specific conversion factor

The default conversion factor in the machine data:

MD10250 $MN_SCALING_VALUE_INCH (conversion factor for switchover to inch system)

is set to 25.4 for converting from the metric to the inch measuring system. By changing the conversion factor, the control system can also be adapted to customer-specific measuring systems.

Axis-specific conversion factor

The default conversion factor in the axis-specific machine data:

MD31200 $MA_SCALING_FACTOR_G70_G71 (conversion factor when G70/G71 is active)

is set to 25.4 for converting from the metric to the inch measuring system. By changing the conversion factor, the control system can also be adapted to customer-specific measuring systems on an axis-specific basis.

6.3.2 Manual switchover of the basic system

General

The control system can operate with the metric or inch measuring system. The initial setting of the measuring system (basic system) is defined using the following machine data:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC (basic system metric).

Depending on the basic system, all length-related data is interpreted either as metric or inch measurements.

The relevant softkey on the HMI in the "Machine" operating area is used to change the measuring system of the controller.

The change in the measuring system occurs only under the following supplementary conditions:

- MD10260 $MN_CONVERT_SCALING_SYSTEM=1
- Bit 0 of MD20110 $MC_RESET_MODE_MASK is set in every channel.
- All channels are in the Reset state.
- Axes do not traverse with JOG, DRF or PLC.
- Constant grinding wheel peripheral speed (GWPS) is not active.

Actions such as part program start or mode change are disabled for the duration of the measuring system changeover.

If the measuring system cannot be changed, this is indicated by a message to that effect on the user interface. These measures ensure that a consistent set of data is always used for a running program with reference to the measuring system.

The actual change in the measuring system is made by writing all the necessary machine data and subsequently activating them with a RESET.

The machine data:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC

and the corresponding G70/G71/G700/G710 settings in the machine data:

MD20150 $MN_GCODE_RESET_VALUES

are automatically switched over consistently for all configured channels.

During this process, the value in machine data:

MD20150 $MC_GCODE_RESET_VALUES[12]

changes between G700 and G710.

This process takes place independently of the protection level currently set.

Note

The availability of the softkey and, therefore, its functionality, can be configured using the compatibility machine data:

MD10260 $MN_CONVERT_SCALING_SYSTEM

If several NCUs are linked by NCU-link, the switchover has the same effect on all linked NCUs. If the requirements for a switchover are not fulfilled on one of the connected NCUs, no switchover will take place on any of the NCUs. It is assumed that interpolations between several NCUs will take place on the existing NCUs, whereby the interpolations can provide correct results only if the same unit systems are used.

References:

Function Manual, Extended Functions; Several Control Panels on Multiple NCUs, Distributed Systems (B3)

System data

When changing over the measuring system, from the view of the user, all length-related specifications are converted to the new measuring system automatically.

This includes:

- Positions
- Feedrates
- Accelerations
- Jerk
- Tool offsets
- Programmable, settable and external zero offsets and DRF offsets
- Compensation values
- Protection zones
- Machine data
- Jog and handwheel factors

After the changeover, all of the above data is available in physical quantities.

Data, for which no unique physical units are defined, is **not** converted automatically.

This includes:

- R parameters
- GUDs (**G**lobal **U**ser **D**ata)
- LUDs (**L**ocal **U**ser **D**ata)
- PUDs (**P**rogram global **U**ser **D**ata)
- Analog inputs/outputs
- Data exchange via FC21

In this case, the user is requested to take into account the currently valid system of units:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC

The current system of units setting can be read using the signal:

DB10 DBX107.7 (inch measuring system)

Using the signal:

DB10 DBB71 (change counter, system of units inch/metric)

the "system of units change counter" can be read out

- The system of units for sag compensation is configured using:

  MD32711 $MA_CEC_SCALING_SYSTEM_METRIC

  References:
  Function Manual Extended Functions; Compensations (K3)

- The measuring system for positional data of the indexing axis tables and switching points for software cams is configured in machine data element:
  MD10270 $MN_POS_TAB_SCALING_SYSTEM.

  References:
  Function Manual, Extended Functions; Software Cams, Limit Switching Signals (N3) / Indexing Axes (T1)

User tool data

For user-defined tool data:

MD18094 $MN_MM_NUM_CC_TDA_PARAM

and tool cutting edge data:

MD18096 $MN_MM_NUM_CC_TOA_PARAM

additional machine data sets are introduced:

MD10290 $MN_CC_TDA_PARAM_UNIT [MM_NUM_CC_TDA_PARAM]

MD10292 $MN_CC_TOA_PARAM_UNIT [MM_NUM_CC_TOA_PARAM]

A physical unit can be configured using these machine data. All length-related user-defined tool data is automatically converted to the new measuring system according to the input on switchover.

Reference point

The reference point is retained. It is not necessary to repeat referencing.

Input resolution and computational resolution

The input/computational resolution is set in the controller using machine data:

MD10200 $MN_INT_INCR_PER_MM

Default settings:

Metric system	Inch system
1000 (0.001 mm)	0.0001

Example:

1 inch = 25.4 mm ⇒ 0.0001 inch = 0.00254 mm = 2.54 µm

To be able to program and display the last 40 mm, MD10200 must be assigned a value of 100000.

Only with this identical setting for both measuring systems is it possible to change the measuring system without a significant loss of accuracy. Once MD10200 has been set to this value, it will not need to be changed each time the measuring system is switched over.

JOG and handwheel factor

The machine data:

MD31090 $MA_JOG_INCR_WEIGHT

consists of two values containing axis-specific increment weighting factors for each of the two measuring systems.

Depending on the actual setting in machine data:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC

the controller automatically sets the appropriate value.

The user defines the two increment factors, e.g. for the first axis, during the installation and startup phase:

- Metric:

  MD31090 $MA_JOG_INCR_WEIGHT[0;AX1]=0.001 mm

- Inch:

  MD31090 $MA_JOG_INCR_WEIGHT[1;AX1]=0.00254 mm ≙ 0.0001 inch

In this way, MD31090 does not have to be written on every inch/metric switchover.

Remaining distances are not accumulated during incremental traversing with JOG when the measuring system is changed, since all internal positions always refer to mm.

Data backup

Data sets which can be separately read out of the controller and have data where the system of units is relevant, receive - when reading - as a function of machine data:

MD10260 $MN_CONVERT_SCALING_SYSTEM

an INCH or METRIC identification corresponding to machine data:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC

This specifies the measuring system in which the data were originally read out.

This information is intended to prevent data sets from being read into the control system with a measuring system, which is different from the active system. In this case, alarm 15030 is triggered and the write process is interrupted.

Since the language instruction is also evaluated in part programs, these can also be "protected" against operator errors as described above. You can, therefore, prevent part programs containing only metric data, for example, from running on an inch measuring system.

Archive and machine data sets are downwards compatible for a setting:

MD11220 $MN_INI_FILE_MODE = 2

Note

The INCH/METRIC operation is only generated if the compatibility machine data:

MD10260 $MN_CONVERT_SCALING_SYSTEM

is set.

Rounding machine data

All length-related machine data is rounded to the nearest 1 pm when writing in the inch measuring system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC=0 and MD10260 $MN_CONVERT_SCALING_SYSTEM=1), in order to avoid rounding problems.

The disturbing loss of accuracy which occurs as a result of conversion to ASCII when reading out a data backup in the inch system of measurement, is corrected by this procedure when the data is read back into the system.

6.3.3 FGROUP and FGREF

Programming

It should be possible to program the effective machining feedrate in the usual way as a path feedrate via the F value in processing procedures where the tool, the workpiece or both are moved by a rotary axis (e.g. laser machining of rotating tubes).

In order to achieve this, it is necessary to specify an effective radius (reference radius) for each of the rotary axes involved. You can do this by programming the modal NC address:
`FGREF[<rotary axis>]=<reference radius>`

The unit of the reference radius depends on the G70/G71/G700/G710 setting.

In order to include the axes in the calculation of the path feedrate, they must all be specified in the FGROUP command.

In order to ensure compatibility with the behavior with no FGREF programming, the evaluation 1 degree = 1 mm is activated on system powerup and RESET.

This corresponds to a reference radius of:

FGREF = 360 mm / (2π) = 57.296 mm

This default is independent of the active basic system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC) and the currently active G70/G71/G700/G710 setting.

Special features of the feedrate weighting for rotary axes in FGROUP:

Program code

```
N100 FGROUP(X,Y,Z,A)
N110 G1 G91 A10 F100
N120 G1 G91 A10 X0.0001 F100
```

The programmed F value in block N110 is evaluated as a rotary axis feedrate in degrees/min, while the feedrate weighting in block N120 is either 100 inch/min or 100 mm/min, depending on the current inch/metric setting.

CAUTION

The FGREF factor also works if only rotary axes are programmed in the block. The normal F value interpretation as degree/min applies in this case only if the radius reference corresponds to the FGREF default:

- For G71/G710: FGREF[A]=57.296
- For G70/G700: FGREF[A]=57.296/25.4

Example

The following example is intended to demonstrate the effect of FGROUP on the path and path feedrate. The variable $AC_TIME contains the time of the block start in seconds. It can only be used in synchronized actions.

```
Program code                    Comment
N100 G0 X0 A0
N110 FGROUP(X,A)
N120 G91 G1 G710 F100           ; Feedrate = 100 mm/min or 100 degrees/min
N130 DO $R1=$AC_TIME
N140 X10                        ; Feedrate = 100 mm/min, path = 10 mm, R1 = approx.
                                  6 s
N150 DO $R2=$AC_TIME
N160 X10 A10                    ; Feedrate = 100 mm/min, path = 14.14 mm, R2
                                  = approx. 8 s
N170 DO $R3=$AC_TIME
N180 A10                        ; Feedrate = 100 degrees/min, path = 10 degrees, R3
                                  = approx. 6 s
N190 DO $R4=$AC_TIME
N200 X0.001 A10                 ; Feedrate = 100 mm/min, path = 10 mm, R4 = approx.
                                  6 s
N210 G700 F100                  ; Feedrate = 2540 mm/min or 100 degrees/min
N220 DO $R5=$AC_TIME
N230 X10                        ; Feedrate = 2540 mm/min, path = 254 mm, R5
                                  = approx. 6 s
N240 DO $R6=$AC_TIME
N250 X10 A10                    ; Feedrate = 2540 mm/min, path = 254.2 mm, R6
                                  = approx. 6 s
N260 DO $R7=$AC_TIME
N270 A10                        ; Feedrate = 100 degrees/min, path = 10 degrees, R7
                                  = approx. 6 s
N280 DO $R8=$AC_TIME
N290 X0.001 A10                 ; Feedrate = 2540 mm/min, path = 10 mm, R8
                                  = approx. 0.288 s
N300 FGREF[A]=360/(2*$PI)       ; Set 1 degree = 1 inch via the effective radius.
N310 DO $R9=$AC_TIME
N320 X0.001 A10                 ; Feedrate = 2540 mm/min, path = 254 mm, R9
                                  = approx. 6 s
N330 M30
```

Diagnostics

Read reference radius

The value of the reference radius of a rotary axis can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variable:

$AA_FGREF[<axis>]	Current main run value

- Without preprocessing stop in the part program via the system variable:

$PA_FGREF[<axis>]	Programmed value

If no values are programmed, the default 360 mm / (2π) = 57.296 mm (corresponding to 1 mm per degree) will be read in both variables.

For linear axes, the value in both variables is always 1 mm.

Read path axes affecting velocity

The axes involved in path interpolation can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variables:

$AA_FGROUP[<axis>]	Returns the value "1" if the specified axis affects the path velocity in the current main run record by means of the basic setting or through FGROUP programming. Otherwise, the variable returns the value "0".
$AC_FGROUP_MASK	Returns a bit key of the channel axes programmed with FGROUP which are to affect the path velocity.

- Without preprocessing stop in the part program via system variables:

$PA_FGROUP[<axis>]	Returns the value "1" if the specified axis affects the path velocity by means of the basic setting or through FGROUP programming. Otherwise, the variable returns the value "0".
$P_FGROUP_MASK	Returns a bit key of the channel axes programmed with FGROUP which are to affect the path velocity.

6.4 Setpoint/actual-value system

6.4.1 General

Control loop

A control loop with the following structure can be configured for every closed-loop controlled axis/spindle:

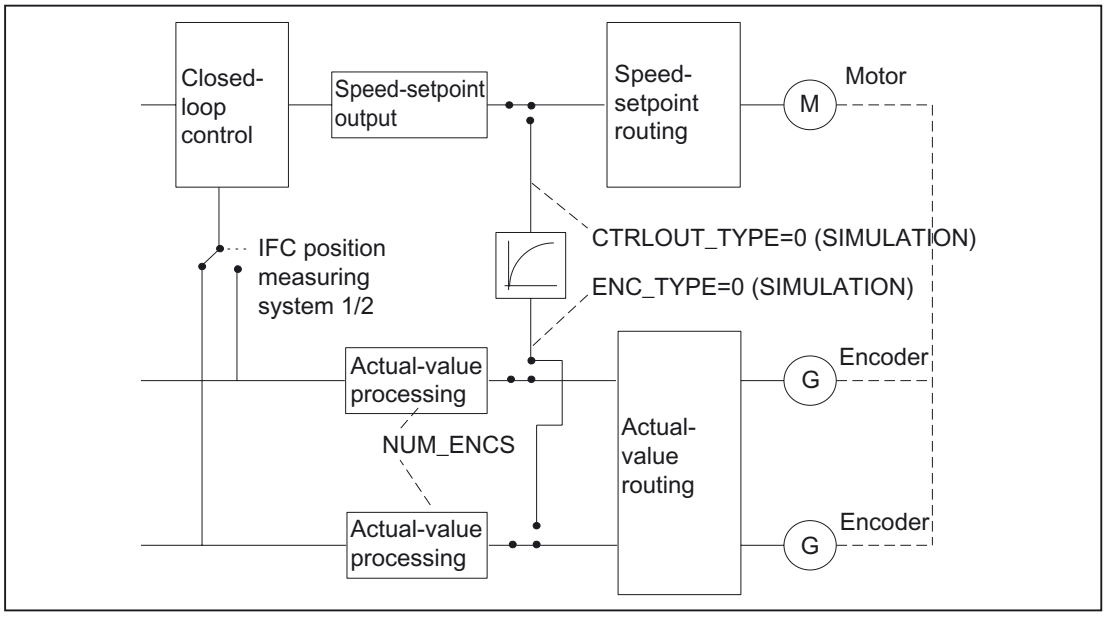


Figure 6-1 Block diagram of a control loop

Setpoint output

A setpoint telegram can be output for each axis/spindle. The setpoint output to the actuator is realized from the SINUMERIK 840D sl.

Actual-value acquisition

A maximum of two measuring systems can be connected for each axis/spindle, e.g. a direct measuring system for machining processes with high accuracy requirements and an indirect measuring system for high-speed positioning tasks.

The number of encoders used is recorded in the machine data:

MD30200 $MA_NUM_ENCS (number of encoders)

In the case of two actual-value branches, the actual value is acquired for both branches.

The active measuring system is always used for position control, absolute value calculation and display. If both measuring systems are activated at the same time by the PLC interface, positioning measuring system 1 is chosen internally by the controller.

Reference point approach is executed by the selected measuring system.

Each position measuring system must be referenced separately.

For an explanation of encoder monitoring, see Section "A3: Axis Monitoring, Protection Zones (Page 85)".

For an explanation of actual-value acquisition compensation functions, see:
References:
Function Manual, Extended Functions; Compensations (K3)

Switching between measuring systems

One can switch between the two measuring systems through the following NC/PLC interface signals:

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

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

For further information, see Section "A2: Various NC/PLC interface signals and functions (Page 33)".

It is possible to switch over measuring systems at any time, the axes do not have to be stationary to do this. Switchover only takes place if a permissible deviation between the actual values and the two measuring systems has not been violated.

The associated tolerance is entered in the machine data:

MD36500 $MA_ENC_CHANGE_TOL (max. tolerance on position actual value switchover)

On switchover, the current difference between position measuring system 1 and 2 is traversed immediately.

Monitoring

The permissible deviation between the actual values of the two measuring systems is to be entered in the machine data:

MD36510 $MA_ENC_DIFF_TOL

For the cyclic comparison of the two measuring systems used, this difference must not be exceeded, as otherwise Alarm 25105 "Measuring systems deviate" is generated.

If the axis is not referenced (at least in the current control measuring system), then the related monitoring is not active if MD36510 = 0 or if neither of the two measuring systems in the axis is active/available.

Types of actual-value acquisition

The used encoder type must be defined through the following machine data:

MD30240 $MA_ENC_TYPE (type of actual-value acquisition (actual position value))

Simulation axes

The speed control loop of an axis can be simulated for test purposes.

The axis "traverses" with a following error, similar to a real axis.

A simulation axis is defined by setting the two following machine data to "0":

MD30130 $MA_CTRLOUT_TYPE[n] (output value of setpoint)

MD30240 $MA_ENC_TYPE[n] (type of actual-value acquisition)

As soon as the standard machine data has been loaded, the axes become simulation axes.

The setpoint and actual value can be set to the reference point value with reference point approach.

The machine data:

MD30350 $MA_SIMU_AX_VDI_OUTPUT (output of axis signals with simulation axes)

can be used to define whether the axis-specific interface signals are to be output on the PLC during the simulation.

Actual-value correction

If actual-value corrections performed by the NC on the encoder selected for position control do not influence the actual value of another encoder defined in the same axis, then this encoder is to be declared as "independent" via the following machine data:

MD30242 $MA_ENC_IS_INDEPENDENT

Actual-value corrections include the following:

- Modulo treatment
- Reference point approach
- Measuring system comparison
- PRESET

6.4.2 Setpoint and encoder assignment

Setpoint marshalling

The following machine data are relevant for the setpoint assignment of a machine axis.

MD30100		$MA_CTRLOUT_SEGMENT_NR[n]
Setpoint assignment, bus segment		
System	Value	Meaning
840D sl	5	PROFIBUS-DP / PROFINET (default)

MD30110		$MA_CTRLOUT_MODULE_NR[n]
Setpoint assignment: Drive number / module number		
System	Value	Meaning
840D sl	x	Index x of MD13050 $MN_DRIVE_LOGIC_ADDRESS[x] should be entered, which refers to the connected drive. MD30110 $MA_CTRLOUT_MODULE_NR[n] = **x**, refers to: MD13050 $MN_DRIVE_LOGIC_ADDRESS[**x**] **Note** The machine data is of no significance if the drive is simulated (MD30130 $MA_CTRLOUT_TYPE[n] = 0).

MD30120		$MA_CTRLOUT_NR[n]
Setpoint assignment: Setpoint output on drive module/module		
System	Value	Meaning
840D sl	1	Modular drive at PROFIBUS / PROFINET with PROFIdrive profile (default)

MD30130		$MA_CTRLOUT_TYPE[n]
Setpoint output type		
System	Value	Meaning
840D sl	0	Simulation (operation without drive)
	1	Setpoint output active

Encoder assignment

The following machine data are relevant for assigning the encoder information of the drive - transferred in the PROFIdrive telegram - to the encoder inputs of the machine axis:

MD30210		$MA_ENC_SEGMENT_NR[n]
Actual value assignment, bus segment		
System	Value	Meaning
840D sl	5	PROFIBUS-DP / PROFINET

MD30220		$MA_ENC_MODULE_NR[n]
Actual value assignment: Drive module number/measuring circuit number		
System	Value	Meaning
840D sl	x	The number of the drive assigned using MD13050 $MN_DRIVE_LOGIC_ADDRESS[x] should be entered. MD30220 $MA_ENC_MODULE_NR[n] = **x**, refers to: MD13050 $MN_DRIVE_LOGIC_ADDRESS[**x**]

MD30230		$MA_ENC_INPUT_NR[n]
Actual value assignment: Input on drive module/measuring circuit module		
System	Value	Meaning
840D sl	x	Number of the encoder interface within the PROFIdrive telegram Examples PROFIdrive telegram 103 x = 1 → 1st encoder interface (G1_ZSW, G1_XIST1, G1_XIST2) x = 2 → 2nd encoder interface (G2_ZSW, G2_XIST1, G2_XIST2) PROFIdrive telegram 118 x = 1 → 1st encoder interface (G2_ZSW, G2_XIST1, G2_XIST2) x = 2 → 2nd encoder interface (G3_ZSW, G3_XIST1, G3_XIST2) **Note** For SINAMICS S120: - encoder 1 (G1_...): Motor encoder - encoder 2 (G2_...): Direct measuring system - encoder 3 (G3_...): Additional measuring system

MD30240		$MA_ENC_TYPE[n]
Encoder type of the actual value sensing (position actual value)		
System	Value	Meaning
840D sl	0	Simulation (operation without encoder)
	1	Incremental encoder
	4	Absolute encoder
Note Corresponds with PROFIdrive parameter p979		

MD30242		$MA_ENC_IS_INDEPENDENT[n, axis]
Encoder is independent		
System	Value	Meaning
840D sl	0	The encoder is not independent.
	1	The encoder is independent. If the actual-value corrections, which are made for the encoder selected for the position control, are not to influence the actual value of the second encoder defined in the same axis, then this should be declared as independent. Actual value corrections are: • - Modulo handling • - Reference point approach • - Measuring system alignment • - PRESET Example: One axis, 2 encoders, the 2nd encoder is independent MD30200 $MA_NUM_ENCS[AX1] = 2 MD30242 $MA_ENC_IS_INDEPENDENT[0, AX1] = 0 MD30242 $MA_ENC_IS_INDEPENDENT[1, AX1] = 1 Selection, position measuring system 1 / 2: DB31.DBX1.5 / 1.6 If encoder 1 is selected for closed-loop position control, then the actual value corrections are only performed on this encoder, as encoder 2 is independent. If encoder 2 is selected for position control, then the actual value corrections are performed on both encoders, as encoder 1 is not independent. This means that the machine data only has an effect on the passive encoder of a machine axis.
	2	The passive encoder is dependent. The encoder actual value is changed by the active encoder. In combination with MD35102 $MA_REFP_SYNC_ENCS = 1, for reference point approach, the passive encoder is aligned to the active encoder - but is NOT referenced. In the referencing mode MD34200 $MA_ENC_REFP_MODE = 3 (distance-coded reference marks) the passive encoder is automatically referenced with the next traversing motion after passing the zero mark distance. This is done independent of the actual operating mode setting.

MD30242		$MA_ENC_IS_INDEPENDENT[n, axis]
Encoder is independent		
System	Value	Meaning
	3	The encoder is independent. For modulo rotary axes, modulo actual value corrections are also performed in the passive encoder.

Note

Machine data index [n]

The machine data index [n] for encoder assignment has the following meaning:

- n = 0: The first encoder assigned to the machine axis
- n = 1: Second encoder assigned to the machine axis

The assignment is made using machine data:

- MD30220$MA_ENC_MODULE_NR[n]
- MD30230$MA_ENC_INPUT_NR[n]

6.4.3 Adapting the motor/load ratios

Gear types

The following gear types are available for adapting the mechanical ratios:

Gear type	Activation	Adaptation	Installation location
Motor/load gear	Parameter set	Fixed configuration	Gear unit
Measuring gear encoder	Power ON	Sensor-dependent	Sensor-side
Load intermediate gear unit	NewConfig	Load-dependent	Tool-side

Local position of gear unit/encoder

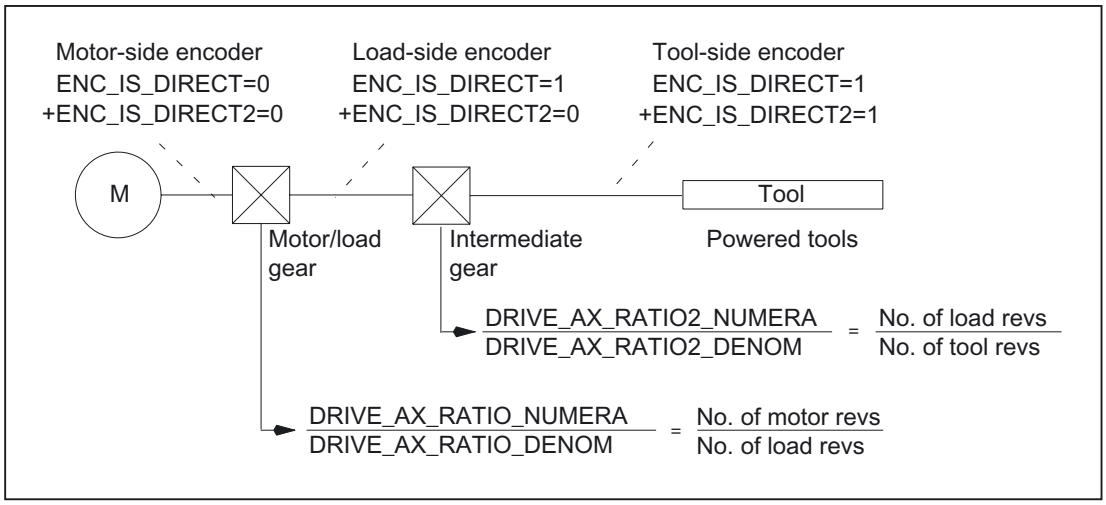


Figure 6-2 Gear unit types and encoder locations

Motor/load gear

The motor/load gear supported by SINUMERIK is configured via the following machine data:

MD31060 $MA_DRIVE_AX_RATIO_NUMERA (numerator load gearbox)

MD31050 $MA_DRIVE_AX_RATIO_DENOM (denominator load gearbox)

The transmission ratio is obtained from the numerator/denominator ratio of both machine data. The associated parameter sets are used automatically as default by the controller to synchronize the position controller with the relevant transmission ratios.

Since a gear stage change is not always carried out automatically, and there are also several ways to request a gear stage change, the position controller is not always incorporated via parameter sets.

Note

For further information about the parameter sets for gear stage change, see Section "S1: Spindles (Page 1383)".

Intermediate gear

Additional, configurable load intermediate gears are also supported by the controller:

MD31066 $MA_DRIVE_AX_RATIO2_NUMERA (intermediate gear numerator)

MD31064 $MA_DRIVE_AX_RATIO2_DENOM (intermediate gear denominator)

Power tools generally have their "own" intermediate gear. Such variable mechanics can be configured by multiplying the active intermediate gearbox and the motor/load gearbox.

⚠ CAUTION

Unlike the motor/load gear, there is no parameter set for the intermediate gear and, therefore, no way of controlling the time-synchronized switchover to the part program or PLC (NC/PLC interface). Part programming during gear change is, therefore, ruled out. It remains the task of the user to match the synchronization of the relevant changed machine data to the corresponding mechanical switchover and activate it. On switchover during a motion, compensations **cannot** be ruled out due to jumps in the scaling factors. These are not monitored for violation of the maximum acceleration.

Encoder directly at the tool

Another connection option is possible for a "tool-side encoder" on the intermediate gear, by configuring machine data:

MD31044 $MA_ENC_IS_DIRECT2

Encoder not directly at the tool

The following supplementary conditions apply to a gear change of the intermediate gear in position-control mode:

- The gear ratio to be changed is incorporated in a re-scaling of the encoder information in this case.

In this case, the following applies to axes/spindles in positioning mode:

- A non-abrupt gear change is **only possible at zero speed**.

  To do this, the tool-side position before and after a gear change are set equal for a change in the ratio, since the mechanical position does not (or hardly) changes during a gear stage change.

  Recommendation:

  To avoid 21612 "Controller enable reset during motion", changeover should be carried out "only at zero speed". It is still permissible and expedient to switch the axis or spindle to speed-control or follow-up mode before or during a gear change.

Supplementary conditions

If the encoder to be used for position control is **connected directly at the tool**, the gear stage change only affects the physical quantities at the speed interface between the NC and the drive of the motor/load gear. The internal parameter sets are not changed.

Reference point and position reference

In the case of gear changes, it is not possible to make a statement about the effect of the reference point or machine position reference on the encoder scaling. In such cases, the controller partially cancels the status "Axis referenced/synchronized".

If the position reference to the machine, tool, etc., has been lost, it must first be restored through appropriate adjustment or referencing of the lost reference point. This is especially important for the functions Travel to fixed stop, Referencing to Bero, Cam and Zero marker.

⚠ CAUTION
The controller cannot detect all possible situations that can lead to loss of the machine position reference. Therefore, it is the responsibility of the commissioning engineer or user to initiate explicit referencing of zero marker synchronization in such cases.

Note

In order to permit new referencing without an interrupting RESET,

machine data:

MD34080 $MA_REFP_MOVE_DIST

and

MD34090 $MA_REFP_MOVE_DIST_CORR

are changed over to NewConfig effectiveness.

For further explanations, see Section "R1: Referencing (Page 1319)".

6.4.4 Speed setpoint output

Control direction and travel direction of the feed axes

You must determine the travel direction of the feed axis before starting work.

Control direction

Before the position control is started up, the speed controller and current controller of the drive must be started up and optimized.

Travel direction

With the machine data:

MD32100 $MA_AX_MOTION_DIR (travel direction),

the direction of motion of the axis can be reversed,

without affecting the control direction of the position control.

Speed setpoint adjustment

SINUMERIK 840D sl

In the case of speed setpoint comparison, the NC is informed which speed setpoint corresponds to which motor speed in the drive, for parameterizing the axial control and monitoring. This comparison is carried out automatically.

For PROFIBUS DP drives, alternatively, the manual speed setpoint comparison is also possible.

- **Manual comparison**

  In the machine data:

  MD32250 $MA_RATED_OUTVAL

  a value not equal to zero is entered.

Note

Velocity adjustment and maximum speed setpoint

Owing to the automatic speed setpoint comparison a velocity adjustment is not necessary for SINUMERIK 840D sl!

Maximum speed setpoint

For SINUMERIK 840D sl, the maximum speed setpoint is defined as a percentage. 100% means maximum speed setpoint or maximum speed for PROFIdrive drives (manufacturer-specific setting parameters in the drive, e.g. p1082 for SINAMICS).

The output of the spindle speed is implemented in the NC for SINUMERIK 840D sl.

Data for five gear stages are realized in the controller.

These stages are defined by a minimum and maximum speed for the stage itself and by a minimum and maximum speed for the automatic gear stage changeover. A new set gear stage is output only if the new programmed speed cannot be traversed in the current gear stage.

With the machine data:

MD36210 $MA_CTRLOUT_LIMIT[n] (maximum speed setpoint)

the speed setpoint is restricted percentage-wise

Values up to 200% are possible.

When the speed is exceeded, an alarm is generated.

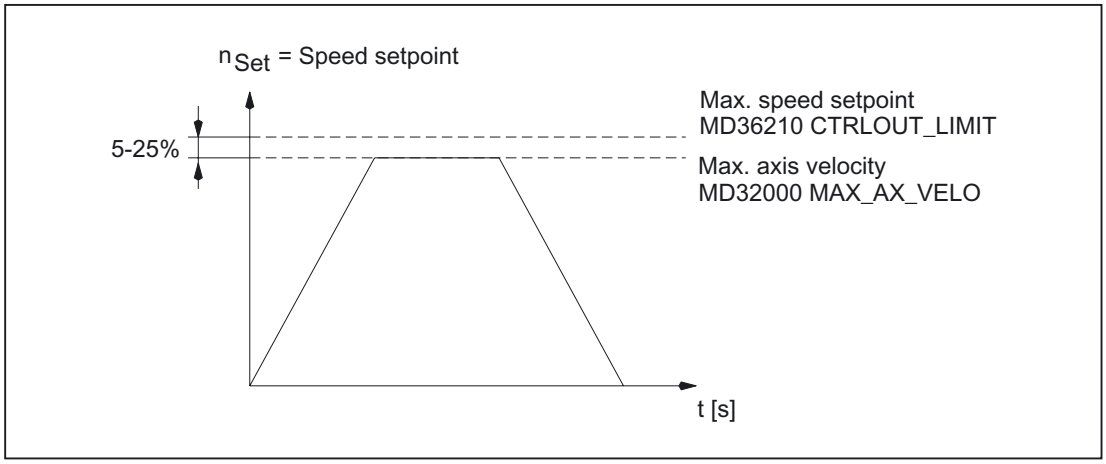


Figure 6-3 Maximum speed setpoint

However, due to control processes, the axes should not reach their maximum velocity (MD32000 $MA_MAX_AX_VELO) at 100% of the speed setpoint, but at 80% to 95%.

In case of axes, whose maximum speed is attained at around 80% of the speed setpoint range, the default value (80%) of the machine data:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

can be taken over.

6.4.5 Actual-value processing

Actual-value resolution

In order to be able to create a correctly closed position control loop, the control system must be informed of the valid actual-value resolution. The following axis-specific machine data is used for this purpose, which is partially specified from the drive (MD31000, 31010, 31020, 31025).

The controller calculates the actual-value resolution based on the machine data. The control parameter sets of the position control are identified as servo parameter sets.

The machining process of the machine forms the basis of the position actual-value acquisition.

Direct measuring system (DM) is on machine directly:	Load-side encoder
Indirect measuring system (IM) is on motor indirectly:	Motor-side encoder

Depending on the type of axis (linear axis, rotary axis) and the type of actual-value acquisition (directly at the machine, indirectly at the motor), the following machine data must be parameterized to calculate the actual-value resolution:

Machine data	Linear axis	Linear axis		Rotary axis	
	Linear scale/ or as direct measuring system	Encoder on motor	Encoder on machine and/or tool	Encoder on motor	Encoder on machine and/or tool
MD30300 $MA_IS_ROT_AX	0	0	0	1	1
MD31000 $MA_ENC_IS_LINEAR[n]	1	0	0	0	0
MD31010 $MA_ENC_GRID_POINT_DIST[n]	Spacing	-	-	-	-
MD34320 $MA_ENC_INVERS[n]	◆	-	-	-	-
MD31040 $MA_ENC_IS_DIRECT[n]	- / 1	0	1	0	1
MD31044 $MA_ENC_IS_DIRECT2[n]	- / 1	0	1	0	1
MD31020 $MA_ENC_RESOL[n]	-	Pulses/ rev	Pulses/ rev	Pulses/r ev	Pulses/ rev
MD31025 $MA_ENC_PULSE_MULT[n]	Encoder multiplication				
MD31030 $MA_LEADSCREW_PITCH	-	mm/rev.	mm/rev.	-	-
MD31050 $MA_DRIVE_AX_RATIO_DENOM[n]	-	Load rev.	-/1	Load rev.	●
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[n]	-	Motor rev. if infeed gear available	-/1	Motor rev.	●

Machine data	Linear axis	Linear axis		Rotary axis	
	Linear scale/ or as direct measuring system	Encoder on motor	Encoder on machine and/or tool	Encoder on motor	Encoder on machine and/or tool
MD31070 $MA_DRIVE_ENC_RATIO_DENOM[n]	-	Encoder rev.	Encoder rev.	Encoder rev.	Encoder rev.
MD31080 $MA_DRIVE_ENC_RATIO_NUMERA[n]	-	Motor-side encoder*	Motor rev.	Motor rev.	Load rev.

- = Does not apply to this combination
- * The encoder on the motor side is a built-in encoder and, therefore, does **not** have a measuring gear unit.
  The transmission ratio is always 1:1.
- ◆ For distance-coded measuring systems
- ● These machine data are not required for encoder matching (path evaluation). However, they must be entered correctly for the setpoint calculation! Otherwise the required servo gain factor (K_V) will not be set. The load revolutions are entered into machine data MD31050 $MA_DRIVE_AX_RATIO_DENOM and the motor revolutions in machine data MD31060 $MA_DRIVE_AX_RATIO_NUMERA.

Machine data

Encoder-dependent machine data	Meaning
MD31000 $MA_ENC_IS_LINEAR[n]	Direct measuring system linear scale
MD31044 $MA_ENC_IS_DIRECT2[n]	Encoder on intermediate gear
MD31070 $MA_DRIVE_ENC_RATIO_DENOM[n]	Measuring gear denominator
MD31080 $MA_DRIVE_ENC_RATIO_NUMERA[n]	Measuring gear numerator
MD31010 $MA_ENC_GRID_POINT_DIST[n]	Distance between reference marks on linear scales
MD31020 $MA_ENC_RESOL[n]	Encoder pulses per revolution for rotary encoder
MD31040 $MA_ENC_IS_DIRECT[n]	Encoder is connected directly at the machine
MD34080 $MA_REFP_MOVE_DIST [n]	Reference point approach distance
MD34090 $MA_REFP_MOVE_DIST_CORR[n]	Reference point offset
MD34320 $MA_ENC_INVERS[n]	Length measuring system is in the opposite sense
n: Encoder index, with n = 0, 1, ... (1st encoder, 2nd encoder, etc.)	

Parameter-set-dependent machine data	Meaning
MD31050 $MA_DRIVE_AX_RATIO_DENOM[m]	Denominator load gearbox
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[m]	Numerator load gearbox
m: Parameter set index, with m = 0, 1, ... (1st parameter set, 2nd parameter set, etc.)	

Encoder and parameter set-independent machine data	Meaning
MD30200 $MA_NUM_ENCS	Number of encoders
MD30300 $MA_IS_ROT_AX	Rotary axis
MD31030 $MA_LEADSCREW_PITCH	Leadscrew pitch
MD31064 $MA_DRIVE_AX_RATIO2_DENOM	Intermediate gear denominator
MD31066 $MA_DRIVE_AX_RATIO2_NUMERA	Intermediate gear numerator
MD32000 $MA_MAX_AX_VELO	Maximum axis velocity

Note

Machine data with NEWCONFIG effectiveness criterion can be activated either in the part program with the command NEWCONF or via the user interface by pressing a softkey.

6.4.6 Actual-value resolution

6.4.6.1 Description of the function

The ratio of control-internal computational resolution to the actual-value resolution is an indication of how exactly the values calculated by the control system can be implemented on the machine.

Linear axes

$$\frac{\text{Computational resolution}}{\text{Actual-value resolution}} = \frac{\text{Internal increments/(mm)}}{\text{Encoder increments/(mm)}}$$

Rotary axes

$$\frac{\text{Computational resolution}}{\text{Actual-value resolution}} = \frac{\text{Internal increments/(degrees)}}{\text{Encoder increments/(degrees)}}$$

Relevant machine data for the actual-value resolution

The actual-value resolution results from the design of the machine, whether gearboxes are available and their gear ratio, the leadscrew pitch for linear axes and the resolution of the encoder being used. The following machine data must be set for this on the control system:

Number	Identifier $MA_	Meaning
30300	IS_ROT_AX	Axis is a rotary axis / spindle
31000	ENC_IS_LINEAR[n]	Measuring system is a linear scale
31010	ENC_GRID_POINT_DIST	Distance between reference marks of the linear scale
31020	ENC_RESOL[n]	Encoder pulses per revolution
31025	ENC_PULSE_MULT[n]	Encoder multiplication (high resolution)
31030	LEADSCREW_PITCH	Leadscrew pitch
31040	ENC_IS_DIRECT[n]	Direct or indirect measuring system
31044	ENC_IS_DIRECT2[n]	Encoders installed at the attached gearbox
31050	DRIVE_AX_RATIO_DENOM[m]	Denominator load gearbox
31060	DRIVE_AX_RATIO_NUMERA[m]	Numerator load gearbox
31064	DRIVE_AX_RATIO2_DENOM	Denominator of attached gearbox
31066	DRIVE_AX_RATIO2_NUMERA	Numerator of attached gearbox
31070	DRIVE_ENC_RATIO_DENOM[n]	Measuring gearbox denominator
31080	DRIVE_ENC_RATIO_NUMERA[n]	Measuring gearbox numerator
n: Encoder index, with n = 0, 1, ... (1st encoder, 2nd encoder, etc.) m: Parameter set index, with m = 0, 1, ... (1st parameter set, 2nd parameter set, etc.)		

Relevant machine data for the computational resolution

The computational resolution, i.e. the resolution with which all the path-related data is calculated in the control system, must be set separately for linear and rotary axes via the following machine data:

Number	Identifier $MA_	Meaning
10200	INT_INCR_PER_MM	Computational resolution for linear positions
10210	INT_INCR_PER_DEG	Computational resolution for angular positions

Recommended setting

The above components and settings that are responsible for the actual-value resolution, should be selected so that the actual-value resolution is higher than the parameterized computational resolution.

$$\frac{\text{Computational resolution}}{\text{Actual-value resolution}} \leq 1$$

6.4.6.2 Example: Linear axis with linear scale

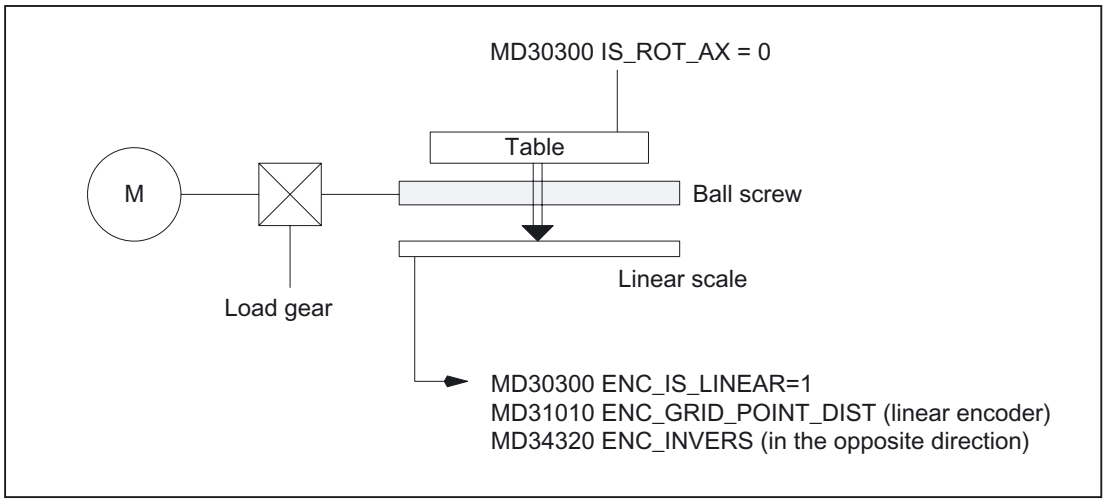


Figure 6-4 Linear axis with linear scale

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

$$\frac{\text{Internal increments / mm}}{\text{Encoder increments /mm}} = \frac{\text{ENC\_GRID\_POINT\_DIST [n] * INT\_INCR\_PER\_MM}}{\text{ENC\_PULSE\_MULT[n]}}$$

6.4.6.3 Example: Linear axis with rotary encoder on motor

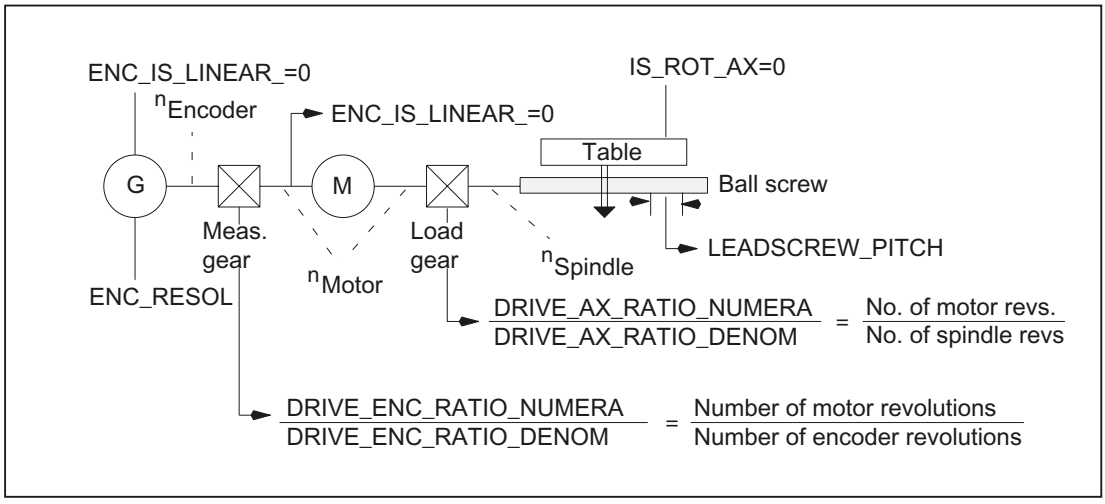


Figure 6-5 Linear axis with rotary encoder on motor

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

$$\frac{\text{Internal increments / mm}}{\text{Encoder increments / mm}} = \frac{1}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \frac{\text{DRIVE\_AX\_RATIO\_DENOM [n]}}{\text{DRIVE\_AX\_RATIO\_NUMERA [n]}} * \text{LEADSCREW\_PITCH} * \text{INT\_INCR\_PER\_MM}$$

Example

Assumptions:

- Rotary encoder on the motor: 2048 pulses/revolution
- Internal pulse multiplication: 2048
- Gearbox, motor / ball screw: 5:1
- Leadscrew pitch: 10 mm/revolution
- Computational resolution: 10000 increments per mm

Machine data	Value
MD30300 $MA_IS_ROT_AX	0
MD31000 $MA_ENC_IS_LINEAR[0]	0
MD31040 $MA_ENC_IS_DIRECT[0]	0
MD31020 $MA_ENC_RESOL[0]	2048
MD31025 $MA_ENC_PULSE_MULT	2048
MD31030 $MA_LEADSCREW_PITCH	10
MD31080 $MA_DRIVE_ENC_RATIO_NUMERA[0]	1
MD31070 $MA_DRIVE_ENC_RATIO_DENOM[0]	1
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[0]	5
MD31050 $MA_DRIVE_AX_RATIO_DENOM[0]	1
MD10210 $MN_INT_INCR_PER_DEG	10000

$$\Rightarrow \frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{1}{2048 * 2048} * \frac{1}{1} * \frac{1}{5}$$

$$* 10\ \text{mm} * 10000\ \text{incr./mm} = 0.004768$$

An encoder increment corresponds to 0.004768 internal increments or 209.731543 encoder increments correspond to an internal increment.

6.4.6.4 Example: Linear axis with rotary encoder on the machine

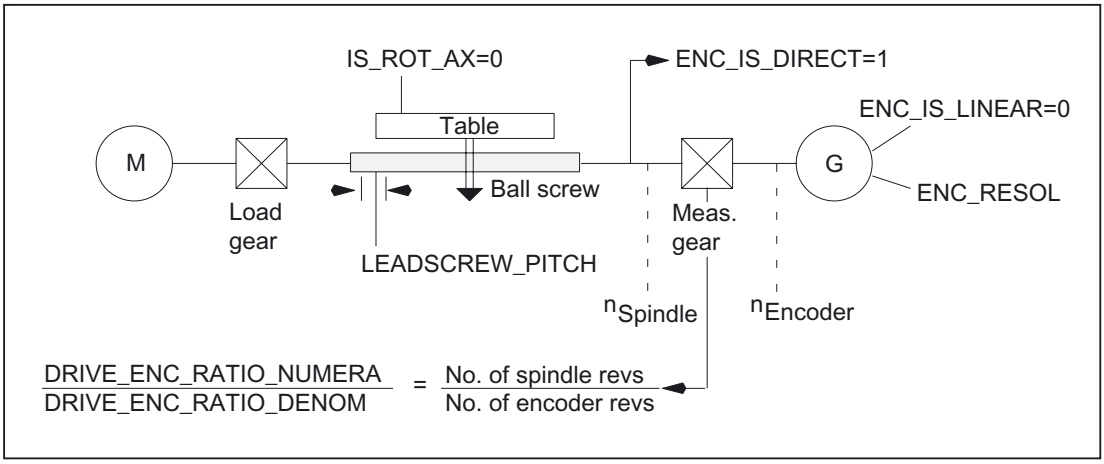


Figure 6-6 Linear axis with rotary encoder on the machine

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

$$\frac{\text{Internal increments / mm}}{\text{Encoder increments / mm}} = \frac{1}{\text{ENC\_RESOL [n]} * \text{ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \text{LEADSCREW\_PITCH} * \text{INT\_INCR\_PER\_MM}$$

6.4.6.5 Example: Rotary axis with rotary encoder on motor

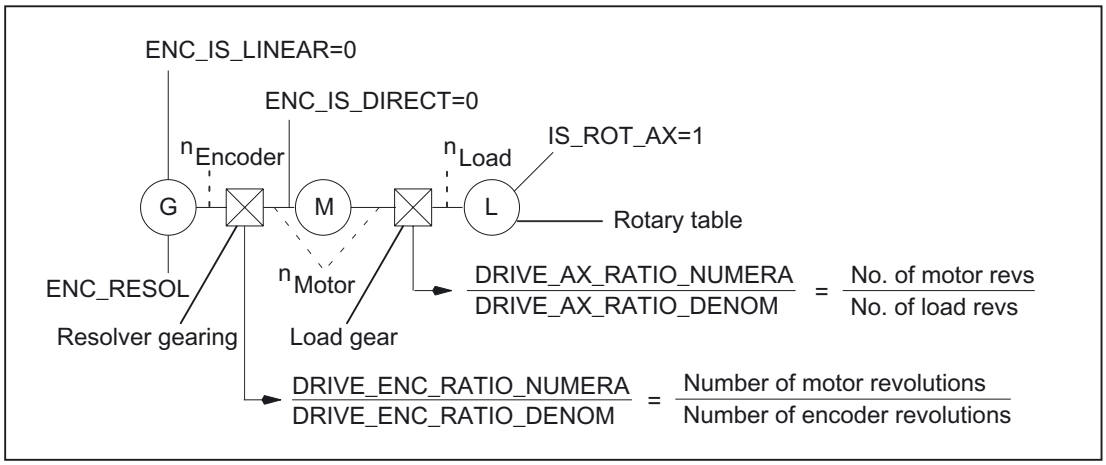


Figure 6-7 Rotary axis with rotary encoder on motor

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

$$\frac{\text{Internal increments / degrees}}{\text{Encoder increments / degrees}} = \frac{360\ \text{degrees}}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \frac{\text{DRIVE\_AX\_RATIO\_DENOM [n]}}{\text{DRIVE\_AX\_RATIO\_NUMERA [n]}} * \text{INT\_INCR\_PER\_DEG}$$

Example

Assumptions:

- Rotary encoder on the motor: 2048 pulses/revolution
- Internal pulse multiplication: 2048
- Gearbox, motor / rotary axis: 5:1
- Computational resolution: 1000 increments per degree

Machine data	Value
MD30300 $MA_IS_ROT_AX	1
MD31000 $MA_ENC_IS_LINEAR[0]	0
MD31040 $MA_ENC_IS_DIRECT[0]	0
MD31020 $MA_ENC_RESOL[0]	2048
MD31025 $MA_ENC_PULSE_MULT	2048
MD31080 $MA_DRIVE_ENC_RATIO_NUMERA[0]	1
MD31070 $MA_DRIVE_ENC_RATIO_DENOM[0]	1
MD31060 $MA_DRIVE_AX_RATIO_NUMERA[0]	5
MD31050 $MA_DRIVE_AX_RATIO_DENOM[0]	1
MD10210 $MN_INT_INCR_PER_DEG	1000

$$\Rightarrow \frac{\text{Internal increments/degrees}}{\text{Encoder increments/degrees}} = \frac{360 \text{ degrees}}{2048 * 2048} * \frac{1}{1} * \frac{1}{5}$$

* 1000 incr./degrees = 0.017166

An encoder increment corresponds to 0.017166 internal increments or 58.254689 encoder increments correspond to an internal increment.

6.4.6.6 Example: Rotary axis with rotary encoder on the machine

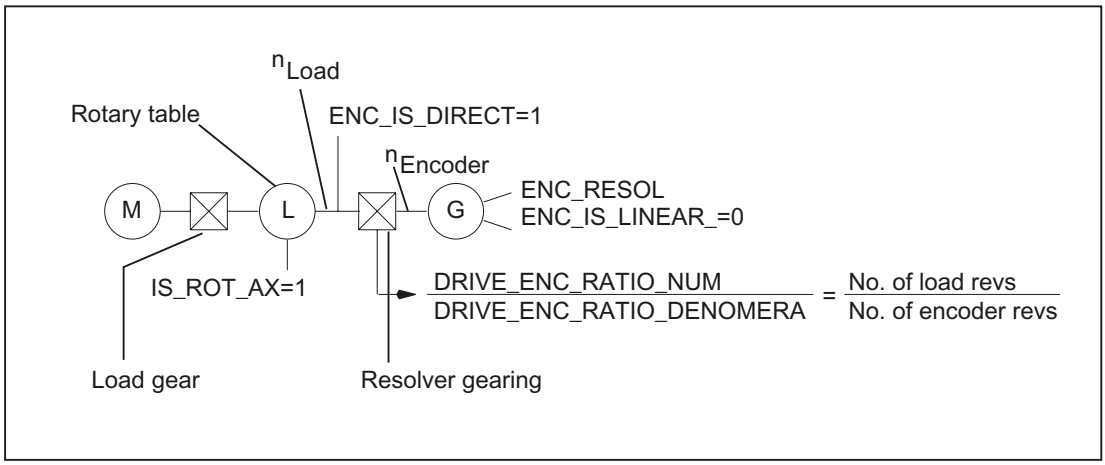


Figure 6-8 Rotary axis with rotary encoder on the machine

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

$$\frac{\text{Internal increments / degrees}}{\text{Encoder increments / degrees}} = \frac{\text{360 degrees}}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \text{INT\_INCR\_PER\_DEG}$$

6.4.6.7 Example: Intermediate gear with encoder on the tool

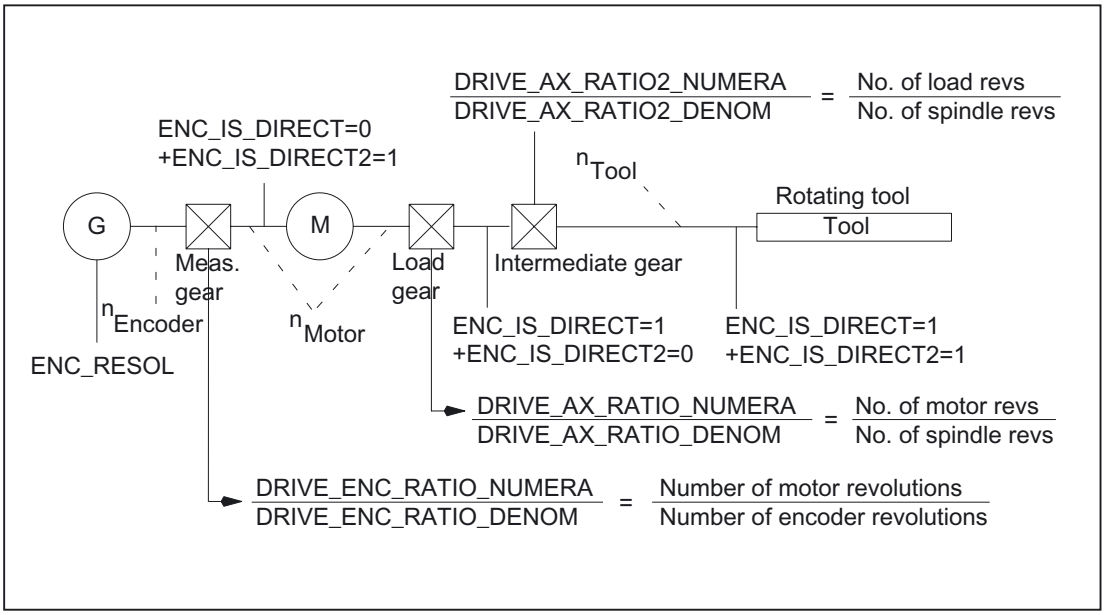


Figure 6-9 Intermediate gear with encoder directly on the rotating tool

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

$$\frac{\text{Internal increments / degrees}}{\text{Encoder increments / degrees}} = \frac{\text{360 degrees}}{\text{ENC\_RESOL [n] * ENC\_PULSE\_MULT[n]}} * \frac{\text{DRIVE\_ENC\_RATIO\_NUMERA [n]}}{\text{DRIVE\_ENC\_RATIO\_DENOM [n]}} * \text{INT\_INCR\_PER\_DEG}$$

6.5 Closed-loop control

6.5.1 General

Position control of an axis/spindle

The closed-loop control of an axis consists of the current and speed control loop of the drive plus a higher-level position control loop in the NC.

The basic structure of an axis/spindle position control is illustrated below:

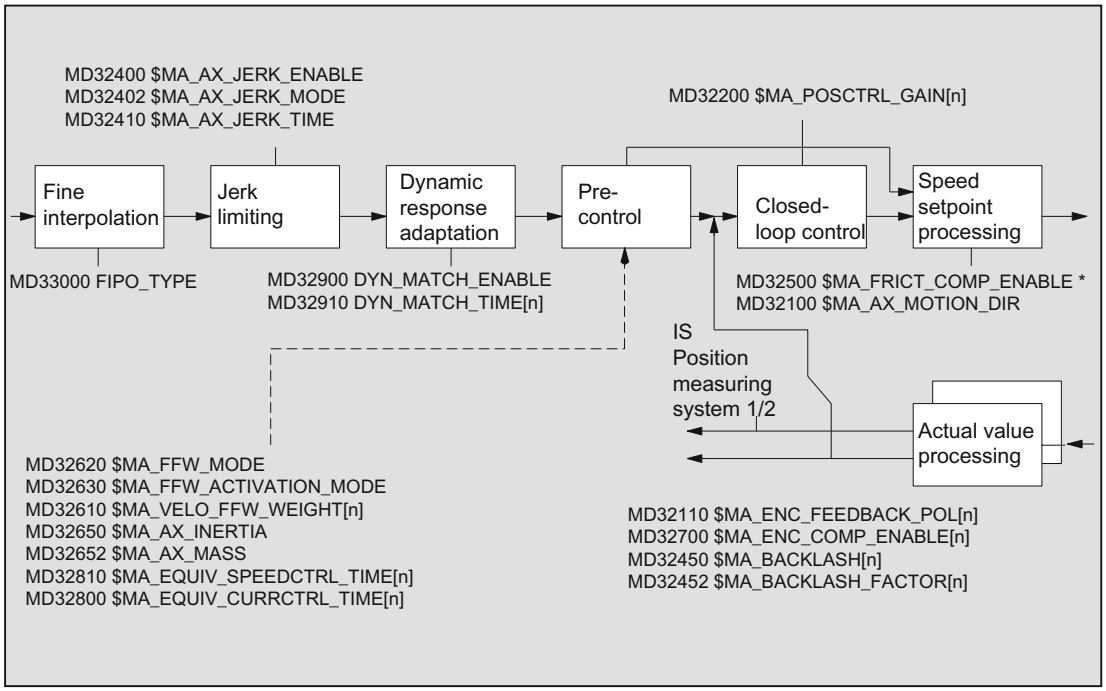


Figure 6-10 Principle representation of the setpoint processing and closed-loop control

For information on the jerk limitation, see Section "B2: Acceleration (Page 245)".

For a description of the feedforward control, backlash, friction compensation with further machine data, and leadscrew error compensation, see:

References:
Function Manual Extended Functions; Compensations (K3)

Fine interpolation

Using the fine interpolator (FIPO), the contour precision can be further increased by reducing the staircase effect in the speed setpoint. You can set three different types of fine interpolation:

MD33000 $MA_FIPO_TYPE = <FIPO mode>

<FIPO mode>	Meaning
1	Differential fine interpolation with mean value generation (smoothing) over an IPO cycle
2	Cubic fine interpolation
3	Cubic fine interpolation optimized for use with the pre-control for the highest contour precision

Servo gain factor

In order that few contour deviations occur in the continuous-path mode, a high servo gain factor (K_v) is necessary:

MD32200 $MA_POSCTRL_GAIN[n]

However, if the servo gain factor (K_v) is too high, instability, overshoot and possibly impermissibly high loads on the machine will result.

The maximum permissible servo gain factor (K_v) depends on the following:

- Design and dynamics of the drive

  (rise time, acceleration and braking capacity)
- Machine quality

  (elasticity, oscillation damping)
- Position control cycle or speed control cycle for active DSC

The servo gain factor (K_v) is defined as follows:

$$KV = \frac{\text{Velocity}}{\text{Following error}} \; ; \; \frac{[\text{m/min}]}{[\text{mm}]}$$

Servo gain factor (K_v) setting for SINUMERIK 840D sl

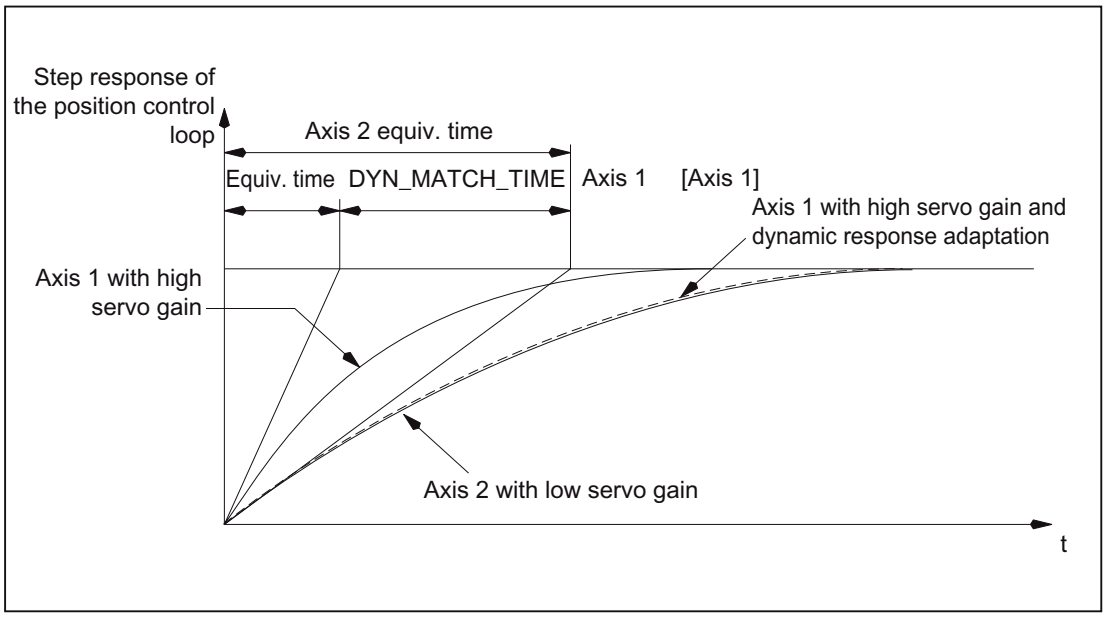


Figure 6-11 Dynamic response adaptation

Dynamic response adaptation

Axes that interpolate with one another, but with different K_v factors can be set to the same following error using the dynamic adaptation function. This allows an optimum contour accuracy to be achieved without loss of control quality by reducing the K_v factors to the dynamically weakest axis.

The function is activated via:

MD32900 $MA_DYN_MATCH_ENABLE = 1 (dynamic response adaptation)

The dynamic response adaptation is realized by entering a new equivalent time constant. It is calculated from the difference in the equivalent time constant of the dynamically weakest axis and the axis to be adapted:

MD32910 $MA_DYN_MATCH_TIME [n] = <difference in the equivalent time constant>

Example of a dynamic response adaptation of three axes without speed feedforward control

The equivalent time constant of the position control loop is:	
Axis 1:	30 ms
Axis 2:	20 ms
Axis 3:	24 ms

With an equivalent time constant of 30 ms, axis 1 is the dynamically weakest axis.

This results in the following new equivalent time constants for the axes:

Axis 1: MD32910 $MA_DYN_MATCH_TIME = 0 ms
Axis 2: MD32910 $MA_DYN_MATCH_TIME = 30 ms - 20 ms = 10 ms
Axis 3: MD32910 $MA_DYN_MATCH_TIME = 30 ms - 24 ms = 6 ms

Approximation formulas for the equivalent time constant of the position control loop of an axis

The equivalent time constant T_{equiv} of the position control loop of an axis is approximately calculated depending on the type of feedforward control:

- Without feedforward control:

$$T_{Spares} \approx \frac{1}{\text{MD32200 POSCTRL\_GAIN [1/s]}}$$

- With speed feedforward control:

$$T_{Spares} \approx \text{MD32810 EQUIV\_SPEEDCTRL\_TIME}$$

- For combined torque/speed feedforward control

$$T_{Spares} \approx \text{MD32800 EQUIV\_CURRCTRL\_TIME}$$

Note

If dynamic response adaptation is realized for a geometry axis, then all other geometry axes must be set to the same dynamic response.

References:
Commissioning Manual CNC: NCK, PLC, drives

6.5.2 Parameter sets of the position controller

The position control

Six parameter sets per machine axis are available to quickly adapt the position control to the changed machine properties during operation, e.g. a gear change of the spindle, or to adjust the dynamic response to another axis, e.g. during tapping.

A parameter set comprises the following machine data:

Number	Identifier $MA_	Meaning
31050	DRIVE_AX_RATIO_DENOM	Denominator load gearbox
31060	DRIVE_AX_RATIO_NUMERA	Numerator load gearbox
32200	POSCTRL_GAIN	Servo gain factor
32452	BACKLASH_FACTOR	Backlash compensation
32610	VELO_FFW_WEIGHT	Feedforward control factor
36012	STOP_LIMIT_FACTOR	Exact stop coarse/fine factor and zero speed
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant, current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant, speed control loop for feed forward control
32910	DYN_MATCH_TIME	Time constant for dynamic response adaptation
36200	AX_VELO_LIMIT	Threshold value for velocity monitoring

Tapping, thread cutting

For tapping / thread cutting, the following applies with regard to the parameter sets of axes:

- For machine axes that are **not** involved in tapping or thread cutting, parameter set 1 (index = 0) is active. The other parameter sets do not have to be taken into account.
- For machine axes that are involved in tapping or thread cutting, the same parameter set number as that of the current gear stage of the spindle is active.

  All parameter sets correspond to the gear stages and must therefore be parameterized.

The current parameter set is displayed on the user interface at:

SINUMERIK Operate

"Operating area switchover" > "Diagnostics" > "Service axis"

Parameter sets during gear stage change

Each gear stage of a spindle is assigned a separate parameter set. The gear stage is selected via the following NC/PLC interface signal:

DB31, ... DBX16.0 - 16.2 = <actual gear stage>

Actual gear stage	DB31, ... DBX16.0 - 16.2	Parameter set	
1st gear stage	000	2	(Index=1)
1st gear stage	001	2	(Index=1)
2nd gear stage	010	3	(Index=2)
3rd gear stage	011	4	(Index=3)
4th gear stage	100	5	(Index=4)
5th gear stage	101 110 111	6	(Index=5)

For further information on gear stages for spindles, see Section "S1: Spindles (Page 1383)".

6.6 Optimization of the control

6.6.1 Position controller, position setpoint filter: Balancing filter

Application

For speed and torque feedforward control

With feedforward control active, the position setpoint is sent through a socalled balancing filter before it reaches the controller itself. It is thus possible to control the speed setpoint to 100% in advance, without resulting in overshoots when positioning.

Filter activation

The filter is activated and the precontrol version selected by changing the axial machine data:

MD32620 $MA_FFW_MODE

Value	Meaning
3	Speed precontrol
4	Combined torque/speed precontrol

Activation of feedforward control

Part programs can be used to activate and deactivate the feedforward control for all axes, using instructions `FFWON` and `FFWOF`, which does not affect the following machine data:

MD32630 $MA_FFW_ACTIVATION_MODE

Control response with POWER ON, RESET, REPOS, etc.

In the case of POWER ON and RESET, as well as with "Enable machine data", the setting data of the feedforward control is read in again (see the appropriate values of the machine data).

Mode change, block search and repositioning have **no** influence on the feedforward control.

Recommended setting in case of recommissioning

If recommissioning, or if default values have been loaded (switch position 1 on commissioning switch and POWER ON), the following machine data default values apply:

MD32620 $MA_FFW_MODE = 3

MD32610 $MA_VELO_FFW_WEIGHT = 1

The balancing time for the speed feedforward control then just has to be adjusted in the following machine data:

MD32810 $MA_EQUIV_SPEEDCTRL_TIME

Setting the equivalent time constant of the speed control loop

MD32810 speed feedforward control

We recommend that the axis be allowed to move in and out in "AUTOMATIC" mode with a part program and that travel-in to the target position, i.e. the actual position value of the active measuring system, be monitored with the servo trace.

The actual position value can also be output to the drive module's digital-to-analog converter, and an oscilloscope can be used for monitoring.

The initial value for setting is the time constant of the speed control loop. This can be read from the reference frequency characteristic of the speed control loop. In the frequent case of a PI controller with speed setpoint smoothing, an approximate equivalent time can be read from drive machine data p1414, p1415, p1416 and p1421.

Another option would be to trace the speed setpoint and actual value at a constant acceleration using an oscilloscope and to measure the follow-on time of the speed actual value.

This start value (e.g. 1.5 ms) is now entered:

MD32810 $MA_EQUIV_SPEEDCTRL_TIME = 0.0015

The axis then travels to and fro and the operator monitors a greatly-magnified characteristic of the position actual value at the target position.

The following rules apply to making manual fine adjustments:

Overshoot monitored:	→ Increase MD32810.
Excessively slow approach monitored:	→ Reduce MD32810.

Increasing MD32810

Increasing the value of MD32810 slows the axis down and increases the geometric contour error on curves to some degree.

It has a similar effect to reducing the position controller gain:

MD32200 $MA_POSCTRL_GAIN

This can also be watched in the Diagnostics area in the screen form "Service Axis" based on the servo gain value calculated.

Reducing MD32810

Reducing the value of MD32810 accelerates the axis.

Therefore, MD32810 should be assigned as small a value as possible, with the overshoot setting the limit during positioning.

MD32810 fine adjustment

Experience has shown that the initial value is only modified slightly during fine adjustment, typically by adding or deducting 0.25 ms.

For example, if the initial value is 1.5 ms, the optimum value calculated manually is usually within the range 1.25 ms to 1.75 ms.

In the case of axes equipped with direct measuring systems (load encoders) and strong elasticity, you may possibly accept small overshoots of some micrometers.

These can be reduced with the help of the position setpoint filter for dynamic response adaptation (MD32910 $MA_DYN_MATCH_TIME) and for jerk (MD32410 $MA_AX_JERK_TIME), which also reduces the axis speed.

Identical axis data within an interpolation group

All the axes within an interpolation group should have **identical settings** in the following data:

MD32200 $MA_POSCTRL_GAIN (adapted using MD32910)

MD32620 $MA_FFW_MODE

MD32610 $MA_VELO_FFW_WEIGHT

MD32810 $MA_EQUIV_SPEEDCTRL_TIME (or MD32800 $MA_EQUIV_CURRCTRL_TIME) (dependent on the mechanical system and drive)

MD32400 $MA_AX_JERK_ENABLE

MD32402 $MA_AX_JERK_MODE

MD32410 $MA_AX_JERK_TIME

The servo gain display (K_V) in the axis service screen form is used for checking.

Unequal axis data of an interpolation group

If identical values are not possible for the above data, the following machine data can be used to make an adjustment:

MD32910 $MA_DYN_MATCH_TIME

This allows the same servo gain value (K_V) to be displayed.

Different servo gain display values (K_V) usually point to the following:

- The gear ratios do not match in one or several axes.
- The feedforward control setting data do not match.

Setting the equivalent time constant of the current control loop

MD32800 Equivalent time constant current control loop for precontrol

The same rules and recommendations apply to setting the time constant of the current control loop as to the speed feedforward control.

However, activation of the torque feedforward control filter with:

MD32620 $MA_FFW_MODE = 4

must, for the purposes of setting the time constant with

MD32800 $MA_EQUIV_CURRCTRL_TIME

but are enabled as before in the drive.

Limitation to stiff machines

Experience has shown that this expenditure is only worthwhile in the case of very stiff machines, and requires appropriate experience. The elasticities of the machine are often excited due to the injection of the torque so strongly that the existing vibrations neutralize the gain in contour accuracy.

In this case, it would be worth trying the use of Dynamic Stiffness Control (DSC) as an alternative:

MD32640 $MA_STIFFNESS_CONTROL_ENABLE = 1

Note

If DSC is to be activated (MD32640 = 1), no parameters may be assigned for actual-value inversion in the NC (MD32110 $MA_ENC_FEEDBACK_POL = -1). Otherwise, error 26017 occurs.

In DSC mode, actual-value inversion may only be undertaken in the drive (SINAMICS parameter p0410).

6.6.2 Position controller, position setpoint filter: Jerk filter

Application

In some applications, such as when milling sculptured surfaces, it can be advantageous to smooth the position setpoint curves to obtain better surfaces, due to reduced excitations of machine vibrations.

Functionality

The filter effect of the position setpoints must be as strong as possible without impermissibly affecting contour accuracy.

The smoothing behavior of the filter must also be as "symmetrical" as possible, i.e. if the same contour was to be traversed both forward and backward, the characteristic rounded by the filter should be as similar as possible in both directions.

The effect of the filter can be monitored by means of the effective servo gain factor (K_V), which is displayed on the Axis service screen form. The filtering effect rounds the position setpoints slightly, thus reducing the path accuracy so that with increasing filter time a smaller effective servo gain factor (K_V) is displayed.

Filter enable with MD32402

The machine data:
MD32400 $MA_AX_JERK_ENABLE
enables the position setpoint filter
which is determined by the following definitions:

MD32402 $MA_AX_JERK_MODE	**= 2**	**Filter mode, moving average value**
MD32410 $MA_AX_JERK_TIME	= 0.02	Set the filter time in seconds (e.g. 20 ms)
MD32400 $MA_AX_JERK_ENABLE	= 1	Enable filter calculation

If filter mode:
MD32402 $MA_AX_JERK_MODE = 2
was not activated previously, then "Power On" must be initiated once.

Otherwise, "Enable machine data" or "Reset" at the machine control panel are sufficient.

Note

Generally, the filter is set using:
MD32402 $MA_AX_JERK_MODE = 2.

Fine adjustment

The fine adjustment of the jerk filter is carried out as follows:

1. Assess the traversing response of the axis
   (e.g. based on positioning processes with servo trace).
2. Modify the filter time in MD32410 $MA_AX_JERK_TIME.
3. Activate the modified time via "Enable machine data" or RESET on the machine control panel.

Disabling

Disabling the jerk filter:

1. Disable filter calculation:
   MD32410 $MA_AX_JERK_ENABLE = 0.
2. Activate the interlock via "Enable machine data" or RESET on the machine control panel.

Supplementary conditions

The jerk filter is available in all controller versions as follows:

- Effective filter times are limited to a range between a minimum of 1 position-control cycle up to a maximum of 32 position-control cycles (31 position-control cycles are available).

Further supplementary conditions regarding the filter effect:

- The display of the calculated servo gain factor (K_V) in the Axis service screen form displays smaller values than would be appropriate based on the filter effect.
- Path accuracy is better than the displayed servo gain (K_V) suggests.

  Therefore, on resetting
  MD32400 $MA_AX_JERK_MODE = 1
  to
  MD32400 $MA_AX_JERK_MODE = 2,
  the displayed servo gain (K_V) can be reduced while retaining the same filter time, although the path accuracy improves.

Axes that interpolate with each other must be set identically.

Once an optimum value has been identified for these axes, the one with the longest filter time should be used as the setting for all axes within the interpolation group.

For further information on jerk limitation at the interpolator level, see Sections "Jerk limitation with single-axis interpolation (SOFTA) (axis-specific) (Page 266)" and "Axis/spindlespecific machine data (Page 300)".

6.6.3 Position control with proportional-plus-integral-action controller

Function

As standard, the core of the position controller is a P controller.

It is possible to switch-in an integral component for special applications (such as an electronic gear). The resulting proportional-plus-integral-action controller then corrects the error between setpoint and actual positions down to zero in a finite, settable time period when the appropriate machine data are set accordingly.

⚠ CAUTION

When the proportional-plus-integral-action controller is active, overshoots occur in the actual position. In this instance, you must decide whether this effect is admissible or acceptable for the application in question. Closed-loop control know-how and measurements using the servo trace are absolutely necessary in order to use the function. If the appropriate machine data are incorrectly set, then machines could be damaged due to instability.

Procedure

1. First optimize the position control loop as a proportional-action controller first using the tools described in the previous subsections.
2. Increase the tolerances of the following machine data while measurements are being taken to determine the quality of the position control with proportional-plus-integral-action controller:
   - MD36020 $MA_POSITIONING_TIME
   - MD36030 $MA_STANDSTILL_POS_TOL
   - MD36040 $MA_STANDSTILL_DELAY_TIME
   - MD36400 $MA_CONTOUR_TOL
3. Activate the position control loop as a proportional-plus-integral-action controller by setting the following machine data:

   MD32220 $MA_POSCTRL_INTEGR_ENABLE ; set value 1

   MD32210 $MA_POSCTRL_INTEGR_TIME ; integral time [sec.]

   Effect of integral time:

   - $T_n \rightarrow 0$:

     The control error is corrected quickly; however, the control loop can become instable.

   - $T_n \rightarrow \infty$:

     The control error is corrected more slowly.
4. If you search for the optimum compromise for T_n.
   T_n for the application between these two extreme cases, you should not select the value to be too close to the instability limit, as otherwise, instability could occur and in turn damage the machine.

5. Use servo trace to trace the travel-in of an automatic program traveling to and from a target position.
6. Set the servo trace to display the following:
   - Following error
   - Actual velocity
   - Actual position
   - Reference position
7. Reset the tolerance values in the following machine data to the required values, once the optimum value for T_n has been identified:
   - MD36020 $MA_POSITIONING_TIME
   - MD36030 $MA_STANDSTILL_POS_TOL
   - MD36040 $MA_STANDSTILL_DELAY_TIME
   - MD36400 $MA_CONTOUR_TOL

Example

Setting result after several iterative processes for $_KR$ and T_n.

Each of the following quantities - following error, actual velocity, actual position, and position setpoint - has been recorded by servo trace. When traversing in JOG mode, the characteristic of the individual data shown in the following figure was then drawn.

Set machine data:

MD32220 $MA_POSCTRL_INTEGR_ENABLE = 1

MD32210 $MA_POSCTRL_INTEGR_TIME = 0.003

MD32200 $MA_POSCTRL_GAIN[1] = 5.0

Parameter set selection 0

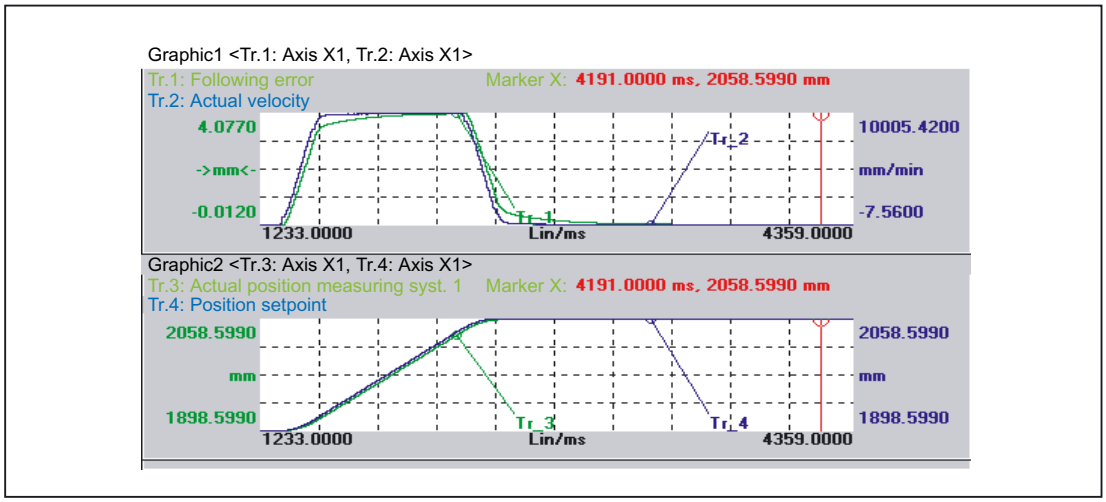


Figure 6-12 Following error (1), actual velocity (2), position actual value (3), position setpoint (4)

6.7 Data lists

6.7.1 Machine data

6.7.1.1 Displaying machine data

Number	Identifier: $MM_	Description
9004	DISPLAY_RESOLUTION	Display resolution
9010	SPIND_DISPLAY_RESOLUTION	Display resolution for spindles
9011	DISPLAY_RESOLUTION_INCH	Display resolution for INCH system of measurement

6.7.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB	Machine axis name
10050	SYSCLOCK_CYCLE_TIME	System basic cycle
10070	IPO_SYSCLOCK_TIME_RATIO	Factor for interpolator cycle
10060	POSCTRL_SYSCLOCK_TIME_RATIO	Factor for position-control cycle
10200	INT_INCR_PER_MM	Computational resolution for linear positions
10210	INT_INCR_PER_DEG	Computational resolution for angular positions
10220	SCALING_USER_DEF_MASK	Activation of scaling factors
10230	SCALING_FACTORS_USER_DEF	Scaling factors of physical quantities
10240	SCALING_SYSTEM_IS_METRIC	Basic system metric
10250	SCALING_VALUE_INCH	Conversion factor for switchover to inch system
10260	CONVERT_SCALING_SYSTEM	Basic system switchover active
10270	POS_TAB_SCALING_SYSTEM	Measuring system of position tables
10290	CC_TDA_PARAM_UNIT	Physical units of the tool data for CC
10292	CC_TOA_PARAM_UNIT	Physical units of the tool edge data for CC
13050	DRIVE_LOGIC_ADDRESS	Logical drive addresses
13060	DRIVE_TELEGRAM_TYPE	Standard message frame type for PROFIBUS DP
13070	DRIVE_FUNCTION_MASK	DP function used
13080	DRIVE_TYPE_DP	Drive type PROFIBUS DP

6.7.1.3 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Initial setting of the G groups

6.7.1.4 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30110	CTRLOUT_MODULE_NR	Setpoint assignment: Drive number
30120	CTRLOUT_NR	Setpoint assignment: Setpoint output on drive module
30130	CTRLOUT_TYPE	Output type of setpoint
30200	NUM_ENCS	Number of encoders
30220	ENC_MODULE_NR	Actual value assignment: Drive module number
30230	ENC_INPUT_NR	Actual value assignment: Input on the drive module
30240	ENC_TYPE	Type of actual-value acquisition (position actual value)
30242	ENC_IS_INDEPENDENT	Encoder is independent
30300	IS_ROT_AX	Rotary axis
31000	ENC_IS_LINEAR	Direct measuring system (linear scale)
31010	ENC_GRID_POINT_DIST	Distance between reference marks on linear scales
31020	ENC_RESOL	Encoder pulses per revolution
31030	LEADSCREW_PITCH	Leadscrew pitch
31040	ENC_IS_DIRECT	Encoder is connected directly to the machine
31044	ENC_IS_DIRECT2	Encoder on intermediate gear
31050	DRIVE_AX_RATIO_DENOM	Denominator load gearbox
31060	DRIVE_AX_RATIO_NUMERA	Numerator load gearbox
31064	DRIVE_AX_RATIO2_DENOM	Intermediate gear denominator
31066	DRIVE_AX_RATIO2_NUMERA	Intermediate gear numerator
31070	DRIVE_ENC_RATIO_DENOM	Measuring gear denominator
31080	DRIVE_ENC_RATIO_NUMERA	Measuring gear numerator
31090	JOG_INCR_WEIGHT	Weighting of increment for INC/handwheel
31200	SCALING_FACTOR_G70_G71	Factor for converting values when G70/G71 is active
32000	MAX_AX_VELO	Maximum axis velocity
32100	AX_MOTION_DIR	Travel direction
32110	ENC_FEEDBACK_POL	Sign actual value (feedback polarity)
32200	POSCTRL_GAIN	Servo gain factor (Kv)
32210	POSCTRL_INTEGR_TIME	Integrator time position controller
32220	POSCTRL_INTEGR_ENABLE	Activation of integral component of position controller
32250	RATED_OUTVAL	Rated output voltage
32260	RATED_VELO	Rated motor speed
32450	BACKLASH	Backlash
32500	FRICT_COMP_ENABLE	Friction compensation active
32610	VELO_FFW_WEIGHT	Feedforward control factor for speed feedforward control
32620	FFW_MODE	Feedforward control mode
32630	FFW_ACTIVATION_MODE	Activate feedforward control from program
32650	AX_INERTIA	Moment of inertia for torque feedforward control
32652	AX_MASS	Axis mass for torque precontrol

Number	Identifier: $MA_	Description
32711	CEC_SCALING_SYSTEM_METRIC	System of measurement of sag compensation
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control loop for feedforward control
32900	DYN_MATCH_ENABLE	Dynamics matching
32910	DYN_MATCH_TIME [n]	Time constant for dynamic response adaptation
32930	POSCTRL_OUT_FILTER_ENABLE	Activation of low-pass filter at position controller output
33000	FIPO_TYPE	Fine interpolator type
34320	ENC_INVERS[n]	Length measuring system is inverse
35100	SPIND_VELO_LIMIT	Maximum spindle speed
36200	AX_VELO_LIMIT [n]	Threshold value for velocity monitoring
36210	CTRLOUT_LIMIT[n]	Maximum speed setpoint
36400	AX_JERK_ENABLE	Axial jerk limitation
36410	AX_JERK_TIME	Time constant for axial jerk filter
36500	ENC_CHANGE_TOL	Max. tolerance for position actual-value switchover
36510	ENC_DIFF_TOL	Measuring system synchronism tolerance
36700	ENC_COMP_ENABLE[n]	Interpolatory compensation

H2: Auxiliary function outputs to PLC

7

7.1 Brief description

7.1.1 Function

Auxiliary functions permit activation of the system functions of the NCK and PLC user functions. Auxiliary functions can be programmed in:

- Part programs
- Synchronized actions
- User cycles

For detailed information on the use of auxiliary function outputs in synchronized actions, see:

References:
Function Manual, Synchronized Actions

Predefined auxiliary functions

Predefined auxiliary functions activate system functions. The auxiliary function is also output to the NC/PLC interface.

The following auxiliary functions are predefined:

Type	Function	Example	Meaning
M	Additional function	M30	End of program
S	Spindle function	S100	Spindle speed 100 (e.g. rpm)
T	Tool number	T3	Tool number 3
D, DL	Tool offset	D1	Tool cutting edge number 1
F	Feedrate	F1000	Feedrate 1000 (e.g. mm/min)

Userdefined auxiliary functions

User-defined auxiliary functions are either extended predefined auxiliary functions or user-specific auxiliary functions.

Extension of predefined auxiliary functions

Extension of predefined auxiliary functions refers to the "address extensions" parameter. The address extension defines the number of the spindle to which the auxiliary function applies. The spindle function M3 (spindle right) is predefined for the master spindle of a channel. If a 2nd spindle is assigned to a channel, a corresponding user-defined auxiliary function must be defined that extends the predefined auxiliary function.

Type	Function	Example	Meaning
M	Additional function	M2=3	2nd spindle: Spindle right
S	Spindle function	S2=100	2nd spindle: Spindle speed = 100 (e.g. rpm)
T	Tool number	T2=3	

User-specific auxiliary functions

User-specific auxiliary functions do not activate system functions. User-specific auxiliary functions are output to the NC/PLC interface only. The functionality of the auxiliary functions must be implemented by the machine manufacturer / user in the PLC user program.

Type	Function	Example	Meaning
H[1)]	Auxiliary function	H2=5	User-specific function

1) Recommendation

7.1.2 Definition of an auxiliary function

An auxiliary function is defined by the following parameters:

- **Type, address extension and value**

  The three parameters are output to the NC/PLC interface.

- **Output behavior**

  The auxiliary function-specific output behavior defines for how long an auxiliary function is output to the NC/PLC interface and when it is output relative to the traversing motion programmed in the same part program block.

- **Group assignment**

  An auxiliary function can be assigned to a particular auxiliary function group. The output behavior can be defined separately for each auxiliary function group. This becomes active if no auxiliary function-specific output behavior has been defined. Group membership also affects output of an auxiliary function after block search.

For more detailed information on auxiliary function output to the NC/PLC interface, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)".

7.1.3 Overview of the auxiliary functions

M functions

M (special function)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
0 (implicit)	- - -	0 ... 99	INT	Function	5
Remarks: The address extension is 0 for the range between 0 and 99. Mandatory without address extension: M0, M1, M2, M17, M30					
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
1 ... 20	Spindle number	1 ... 99	INT	Function	5
Remarks: M3, M4, M5, M19, M70 with address extension as the spindle number. (e.g. M2=5; spindle stop for spindle 2). Without an address extension, the function influences the master spindle.					
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
0 ... 99	Any	100 ... 2147483647	INT	Function	5
Remarks: User-specific M functions.					

[8] See "Meaning of footnotes" at the end of the overview.

Use

Controlling machine functions in synchronism with the part program.

Further information

- The following M functions have a predefined meaning: M0, M1, M2, M17, M30

  M3, M4, M5, M6, M19, M70, M40, M41, M42, M43, M44, M45.
- For each M function (M0 - M99), there is a dynamic signal at the NC/PLC interface that indicates the validity (new output) of the M function. In addition, 64 further signals can be assigned for user M functions (see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)").
- For subprograms, machine data can be used to set whether an output of the M function should be undertaken for the end of the part program M17, M2 and M30 to the PLC:

  MD20800 $MC_SPF_END_TO_VDI (subprogram end to PLC)
- For the predefined M function M40 – M45, only limited redefinition of the output specification is possible.

- The predefined auxiliary functions M0, M1, M17, M30, M6, M4, M5 cannot be redefined.
- M function-specific machine data:

  MD10800 $MN_EXTERN_CHAN_SYNC_M_NO_MIN

  MD10802 $MN_EXTERN_CHAN_SYNC_M_NO_MAX

  MD10804 $MN_EXTERN_M_NO_SET_INT

  MD10806 $MN_EXTERN_M_NO_DISABLE_INT

  MD10814 $MN_EXTERN_M_NO_MAC_CYCLE

  MD10815 $MN_EXTERN_M_NO_MAC_CYCLE_NAME

  MD20094 $MC_SPIND_RIGID_TAPPING_M_NR

  MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR

  MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO

  MD22200 $MC_AUXFU_M_SYNC_TYPE

  MD22530 $MC_TOCARR_CHANGE_M_CODE

  MD22532 $MC_GEOAX_CHANGE_M_CODE

  MD22534 $MC_TRAFO_CHANGE_M_CODE

  MD22560 $MC_TOOL_CHANGE_M_CODE

S functions

S (spindle function)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
0 ... 20	Spindle number [5]	0 ... ± 3.4028 exp38 [3]	REAL	Spindle speed	3
Remarks: The master spindle of the channel is addressed if no address extension is specified.					

[3] , [5], [8]See "Meaning of footnotes" at the end of the overview.

Use

Spindle speed.

Further information

- S functions are assigned to auxiliary function group 3 by default.
- Without an address extension, the S functions refer to the master spindle of the channel.
- S function-specific machine data:

  MD22210 $MC_AUXFU_S_SYNC_TYPE (output time of the S functions)

H functions

H (aux. function)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
0 ... 99	Any	- 2147483648 ... + 2147483647	INT	Any	3
		0 ... ± 3.4028 exp38 [2] [3] [4]	REAL		
Remarks: The functionality must be implemented by the user in the PLC user program.					

[2] [3] [4] [8] See "Meaning of footnotes" at the end of the overview.

Use

User-specific auxiliary functions.

Further information

- H function-specific machine data:

  MD22110 $MC_AUXFU_H_TYPE_INT (type of H-auxiliary function is an integer)

  MD22230 $MC_AUXFU_H_SYNC_TYPE (output time of the H functions)

T functions

T (tool number) [5] [6]					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
1 ... 12	Spindle number (with active tool management)	0 ... 32000 (also symbolic tool names for active tool management)	INT	Selection of the tool	1
Remarks: Tool names are not output to the PLC. [1]					

[1] [5] [6] [8] See "Meaning of footnotes" at the end of the overview.

Use

Tool selection.

Further information

- Identification of the tools, optionally via tool number or location number (see Section "W1: Tool offset (Page 1567)").

  References:
  Function Manual Tool Management
- When T0 is selected, the current tool is removed from the toolholder but not replaced by a new tool (default setting).
- T function-specific machine data:

  MD22220 $MC_AUXFU_T_SYNC_TYPE (output time of the T functions)

D functions

D (tool offset)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
- - -	- - -	0 ... 9	INT	Selection of the tool offset	1
Remarks: Clearing the tool offset with D0. Default is D1.					

[8] See "Meaning of footnotes" at the end of the overview.

Use

Selection of the tool offset.

Further information

- Initial setting: D1
- After a tool change, the default tool cutting edge can be parameterized via:

  MD20270 $MC_CUTTING_EDGE_DEFAULT (basic position of the tool cutting edge without programming)
- Deselection of the tool offset: D0
- D function-specific machine data:

  MD22250 $MC_AUXFU_D_SYNC_TYPE (output time of the D functions)

DL functions

DL (additive tool offset)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
- - -	- - -	0 ... 6	INT	Selection of the additive tool offset	1
Remarks: The additive tool offset selected with DL refers to the active D number.					

[8] See "Meaning of footnotes" at the end of the overview.

Use

Selection of the additive tool offset with reference to an active tool offset.

Further information

- Initial setting: DL = 0
- DL values cannot be output to the PLC via synchronized actions.
- Default setting of the additive tool offset without an active DL function:

  MD20272 $MC_SUMCORR_DEFAULT (basic setting of the additive offset without a program)
- Deselection of the additive tool offset: DL = 0
- DL function-specific machine data:

  MD22252 $MC_AUXFU_DL_SYNC_TYPE (output time DL functions)

F functions

F (feedrate)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
- - -	- - -	0.001 ... 999 999.999	REAL	Path feedrate	6
Remarks: - - -					

[8] See "Meaning of footnotes" at the end of the overview.

Use

Path velocity.

Further information

- F function-specific machine data:

  MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

FA functions

FA (axial feedrate)					
Address extension		**Value**			
Range of values	**Meaning**	**Range of values**	**Type**	**Meaning**	**Number** [8]
1 - 31	Axis number	0.001 ... 999 999.999	REAL	Axial feedrate	6
Remarks: - - -					

[8] See "Meaning of footnotes" at the end of the overview.

Use

Axial velocity.

Further information

- F function-specific machine data:

  MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

Meaning of footnotes

1) If tool management is active, neither a T change signal nor a T word is output to the interface (channel).

2) The type for the values can be selected by the user via MD22110 $MC_AUXFU_H_TYPE_INT.

3) Because of the limited display options on the operator panel screens, the REAL type values displayed are restricted to:

   –999 999 999.9999 to 999 999 999.9999

   The NC calculates internally but with complete accuracy.

4) The REAL values are rounded and output to the PLC when setting the machine data:

   MD22110 $MC_AUXFU_H_TYPE_INT = 1 (type of H-auxiliary functions is an integer)

   The PLC user program must interpret the value transferred according to the machine data setting.

5) If the tool management is active, the meaning of the address extension can be parameterized. Address extension = 0 means the value must be replaced by that of the master spindle number, i.e. it is equivalent to not programming the address extension.

   Auxiliary functions M19 "Position spindle" collected during a block search are not output to the PLC.

6) M6: Range of values of the address extension:

   - without tool management: 0 ... 99

   - with tool management: 0 ... maximum spindle number

   0: to be replaced by the value of the master spindle number or master toolholder

7) If tool management is active, the auxiliary function M6 "Tool change" can only be programmed once in a part program block, irrespective the address extensions that are programmed.

8) Maximum number of auxiliary functions per part program block.

7.2 Predefined auxiliary functions

Function

Every pre-defined auxiliary function is assigned to a system function and cannot be changed. If a pre-defined auxiliary function is programmed in a part program/cycle, then this is output to the PLC via the NC/PLC interface and the corresponding system function is executed in the NCK.

Definition of a predefined auxiliary function

The parameters of the predefined auxiliary function are stored in machine data and can be changed in some cases. All machine data, which are assigned to an auxiliary function, have the same index <n>.

- MD22040 $MC_AUXFU_PREDEF_GROUP[<n>] (group assignment of predefined auxiliary functions)
- MD22050 $MC_AUXFU_PREDEF_TYPE[<n>] (type of predefined auxiliary functions)
- MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>] (address extension for predefined auxiliary functions)
- MD22070 $MC_AUXFU_PREDEF_VALUE[<n>] (value of predefined auxiliary functions)
- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>] (output behavior of predefined auxiliary functions)

7.2.1 Overview: Predefined auxiliary functions

Significance of the parameters listed in the following tables:

Parameter	Meaning
Index <n>	Machine data index of the parameters of an auxiliary function
Type	MD22050 $MC_AUXFU_PREDEF_TYPE[<n>]
Address extension	MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>]
Value	MD22070 $MC_AUXFU_PREDEF_VALUE[<n>]
Group	MD22040 $MC_AUXFU_PREDEF_GROUP[<n>]

Predefined auxiliary functions

General auxiliary functions, Part 1					
System function	Index <n>	Type	Address extension	Value	Group
Stop	0	M	0	0	1
Conditional stop	1	M	0	1	1
End of subprogram	2	M	0	2	1
	3	M	0	17	1
	4	M	0	30	1
Tool change	5	M	(0)	6 [1)]	(1)

Spindle-specific auxiliary functions, spindle 1					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	6	M	1	3	(2)
Spindle left	7	M	1	4	(2)
Spindle stop	8	M	1	5	(2)
Position spindle	9	M	1	19	(2)
Axis mode	10	M	1	70 [2)]	(2)
Automatic gear stage	11	M	1	40	(4)
Gear stage 1	12	M	1	41	(4)
Gear stage 2	13	M	1	42	(4)
Gear stage 3	14	M	1	43	(4)
Gear stage 4	15	M	1	44	(4)
Gear stage 5	16	M	1	45	(4)
Spindle speed	17	S	1	-1	(3)

General auxiliary functions, Part 2					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Feedrate	18	F	0	-1	(1)
Cutting edge selection	19	D	0	-1	(1)
DL	20	L	0	-1	(1)
Tool selection	21	T	(0)	-1	(1)
Stop (associated)	22	M	0	-1 3)	1
Conditional stop (associated)	23	M	0	-1 4)	1
End of subprogram	24	M	0	-1 5)	1
Nibbling	25	M	0	20 6)	(10)
Nibbling	26	M	0	23 6)	(10)
Nibbling	27	M	0	22 6)	(11)
Nibbling	28	M	0	25 6)	(11)
Nibbling	29	M	0	26 6)	(12)
Nibbling	30	M	0	122 6)	(11)
Nibbling	31	M	0	125 6)	(11)
Nibbling	32	M	0	27 6)	(12)

Spindle-specific auxiliary functions, spindle 2					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	33	M	2	3	(72)
Spindle left	34	M	2	4	(72)
Spindle stop	35	M	2	5	(72)
Position spindle	36	M	2	19	(72)
Axis mode	37	M	2	70 2)	(72)
Automatic gear stage	38	M	2	40	(74)
Gear stage 1	39	M	2	41	(74)
Gear stage 2	40	M	2	42	(74)
Gear stage 3	41	M	2	43	(74)
Gear stage 4	42	M	2	44	(74)
Gear stage 5	43	M	2	45	(74)
Spindle speed	44	S	2	-1	(73)

Spindle-specific auxiliary functions, spindle 3					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	45	M	3	3	(75)
Spindle left	46	M	3	4	(75)
Spindle stop	47	M	3	5	(75)
Position spindle	48	M	3	19	(75)
Axis mode	49	M	3	70 2)	(75)
Automatic gear stage	50	M	3	40	(77)
Gear stage 1	51	M	3	41	(77)
Gear stage 2	52	M	3	42	(77)
Gear stage 3	53	M	3	43	(77)
Gear stage 4	54	M	3	44	(77)
Gear stage 5	55	M	3	45	(77)
Spindle speed	56	S	3	-1	(76)

Spindle-specific auxiliary functions, spindle 4					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	57	M	4	3	(78)
Spindle left	58	M	4	4	(78)
Spindle stop	59	M	4	5	(78)
Position spindle	60	M	4	19	(78)
Axis mode	61	M	4	70 2)	(78)
Automatic gear stage	62	M	4	40	(80)
Gear stage 1	63	M	4	41	(80)
Gear stage 2	64	M	4	42	(80)
Gear stage 3	65	M	4	43	(80)
Gear stage 4	66	M	4	44	(80)
Gear stage 5	67	M	4	45	(80)
Spindle speed	68	S	4	-1	(79)

Spindle-specific auxiliary functions, spindle 5					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	69	M	5	3	(81)
Spindle left	70	M	5	4	(81)
Spindle stop	71	M	5	5	(81)
Position spindle	72	M	5	19	(81)
Axis mode	73	M	5	70 [2)]	(81)
Automatic gear stage	74	M	5	40	(83)
Gear stage 1	75	M	5	41	(83)
Gear stage 2	76	M	5	42	(83)
Gear stage 3	77	M	5	43	(83)
Gear stage 4	78	M	5	44	(83)
Gear stage 5	79	M	5	45	(83)
Spindle speed	80	S	5	-1	(82)

Spindle-specific auxiliary functions, spindle 6					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	81	M	6	3	(84)
Spindle left	82	M	6	4	(84)
Spindle stop	83	M	6	5	(84)
Position spindle	84	M	6	19	(84)
Axis mode	85	M	6	70 [2)]	(84)
Automatic gear stage	86	M	6	40	(86)
Gear stage 1	87	M	6	41	(86)
Gear stage 2	88	M	6	42	(86)
Gear stage 3	89	M	6	43	(86)
Gear stage 4	90	M	6	44	(86)
Gear stage 5	91	M	6	45	(86)
Spindle speed	92	S	6	-1	(85)

Spindle-specific auxiliary functions, spindle 7					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	93	M	7	3	(87)
Spindle left	94	M	7	4	(87)
Spindle stop	95	M	7	5	(87)
Position spindle	96	M	7	19	(87)
Axis mode	97	M	7	70 2)	(87)
Automatic gear stage	98	M	7	40	(89)
Gear stage 1	99	M	7	41	(89)
Gear stage 2	100	M	7	42	(89)
Gear stage 3	101	M	7	43	(89)
Gear stage 4	102	M	7	44	(89)
Gear stage 5	103	M	7	45	(89)
Spindle speed	104	S	7	-1	(88)

Spindle-specific auxiliary functions, spindle 8					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	105	M	8	3	(90)
Spindle left	106	M	8	4	(90)
Spindle stop	107	M	8	5	(90)
Position spindle	108	M	8	19	(90)
Axis mode	109	M	8	70 2)	(90)
Automatic gear stage	110	M	8	40	(92)
Gear stage 1	111	M	8	41	(92)
Gear stage 2	112	M	8	42	(92)
Gear stage 3	113	M	8	43	(92)
Gear stage 4	114	M	8	44	(92)
Gear stage 5	115	M	8	45	(92)
Spindle speed	116	S	8	-1	(91)

Spindle-specific auxiliary functions, spindle 9					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	117	M	9	3	(93)
Spindle left	118	M	9	4	(93)
Spindle stop	119	M	9	5	(93)
Position spindle	120	M	9	19	(93)
Axis mode	121	M	9	70 [2)]	(93)
Automatic gear stage	122	M	9	40	(95)
Gear stage 1	123	M	9	41	(95)
Gear stage 2	124	M	9	42	(95)
Gear stage 3	125	M	9	43	(95)
Gear stage 4	126	M	9	44	(95)
Gear stage 5	127	M	9	45	(95)
Spindle speed	128	S	9	-1	(94)

Spindle-specific auxiliary functions, spindle 10					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	129	M	10	3	(96)
Spindle left	130	M	10	4	(96)
Spindle stop	131	M	10	5	(96)
Position spindle	132	M	10	19	(96)
Axis mode	133	M	10	70 [2)]	(96)
Automatic gear stage	134	M	10	40	(98)
Gear stage 1	135	M	10	41	(98)
Gear stage 2	136	M	10	42	(98)
Gear stage 3	137	M	10	43	(98)
Gear stage 4	138	M	10	44	(98)
Gear stage 5	139	M	10	45	(98)
Spindle speed	140	S	10	-1	(97)

Spindle-specific auxiliary functions, spindle 11					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	141	M	11	3	(99)
Spindle left	142	M	11	4	(99)
Spindle stop	143	M	11	5	(99)
Position spindle	144	M	11	19	(99)
Axis mode	145	M	11	70 [2)]	(99)
Automatic gear stage	146	M	11	40	(101)
Gear stage 1	147	M	11	41	(101)
Gear stage 2	148	M	11	42	(101)
Gear stage 3	149	M	11	43	(101)
Gear stage 4	150	M	11	44	(101)
Gear stage 5	151	M	11	45	(101)
Spindle speed	152	S	11	-1	(100)

Spindle-specific auxiliary functions, spindle 12					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	153	M	11	3	(102)
Spindle left	154	M	12	4	(102)
Spindle stop	155	M	12	5	(102)
Position spindle	156	M	12	19	(102)
Axis mode	157	M	12	70 [2)]	(102)
Automatic gear stage	158	M	12	40	(104)
Gear stage 1	159	M	12	41	(104)
Gear stage 2	160	M	12	42	(104)
Gear stage 3	161	M	12	43	(104)
Gear stage 4	162	M	12	44	(104)
Gear stage 5	163	M	12	45	(104)
Spindle speed	164	S	12	-1	(103)

Spindle-specific auxiliary functions, spindle 13					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	165	M	13	3	(105)
Spindle left	166	M	13	4	(105)
Spindle stop	167	M	13	5	(105)
Position spindle	168	M	13	19	(105)
Axis mode	169	M	13	70 [2)]	(105)
Automatic gear stage	170	M	13	40	(107)
Gear stage 1	171	M	13	41	(107)
Gear stage 2	172	M	13	42	(107)
Gear stage 3	173	M	13	43	(107)
Gear stage 4	174	M	13	44	(107)
Gear stage 5	175	M	13	45	(107)
Spindle speed	176	S	13	-1	(106)

Spindle-specific auxiliary functions, spindle 14					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	177	M	14	3	(108)
Spindle left	178	M	14	4	(108)
Spindle stop	179	M	14	5	(108)
Position spindle	180	M	14	19	(108)
Axis mode	181	M	14	70 [2)]	(108)
Automatic gear stage	182	M	14	40	(110)
Gear stage 1	183	M	14	41	(110)
Gear stage 2	184	M	14	42	(110)
Gear stage 3	185	M	14	43	(110)
Gear stage 4	186	M	14	44	(110)
Gear stage 5	187	M	14	45	(110)
Spindle speed	188	S	14	-1	(109)

Spindle-specific auxiliary functions, spindle 15					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	189	M	15	3	(111)
Spindle left	190	M	15	4	(111)
Spindle stop	191	M	15	5	(111)
Position spindle	192	M	15	19	(111)
Axis mode	193	M	15	70 [2)]	(111)
Automatic gear stage	194	M	15	40	(113)
Gear stage 1	195	M	15	41	(113)
Gear stage 2	196	M	15	42	(113)
Gear stage 3	197	M	15	43	(113)
Gear stage 4	198	M	15	44	(113)
Gear stage 5	199	M	15	45	(113)
Spindle speed	200	S	15	-1	(112)

Spindle-specific auxiliary functions, spindle 16					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	201	M	16	3	(114)
Spindle left	202	M	16	4	(114)
Spindle stop	203	M	16	5	(114)
Position spindle	204	M	16	19	(114)
Axis mode	205	M	16	70 [2)]	(114)
Automatic gear stage	206	M	16	40	(116)
Gear stage 1	207	M	16	41	(116)
Gear stage 2	208	M	16	42	(116)
Gear stage 3	209	M	16	43	(116)
Gear stage 4	210	M	16	44	(116)
Gear stage 5	211	M	16	45	(116)
Spindle speed	212	S	16	-1	(115)

Spindle-specific auxiliary functions, spindle 17					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	213	M	17	3	(117)
Spindle left	214	M	17	4	(117)
Spindle stop	215	M	17	5	(117)
Position spindle	216	M	17	19	(117)
Axis mode	217	M	17	70 [2)]	(117)
Automatic gear stage	218	M	17	40	(119)
Gear stage 1	219	M	17	41	(119)
Gear stage 2	220	M	17	42	(119)
Gear stage 3	221	M	17	43	(119)
Gear stage 4	222	M	17	44	(119)
Gear stage 5	223	M	17	45	(119)
Spindle speed	224	S	17	-1	(118)

Spindle-specific auxiliary functions, spindle 18					
System function	**Index <n>**	**Type**	**Address extension**	**Value**	**Group**
Spindle right	225	M	18	3	(120)
Spindle left	226	M	18	4	(120)
Spindle stop	227	M	18	5	(120)
Position spindle	228	M	18	19	(120)
Axis mode	229	M	18	70 [2)]	(120)
Automatic gear stage	230	M	18	40	(122)
Gear stage 1	231	M	18	41	(122)
Gear stage 2	232	M	18	42	(122)
Gear stage 3	233	M	18	43	(122)
Gear stage 4	234	M	18	44	(122)
Gear stage 5	235	M	18	45	(122)
Spindle speed	236	S	18	-1	(121)

Spindle-specific auxiliary functions, spindle 19					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	237	M	19	3	(123)
Spindle left	238	M	19	4	(123)
Spindle stop	239	M	19	5	(123)
Position spindle	240	M	19	19	(123)
Axis mode	241	M	19	70 [2)]	(123)
Automatic gear stage	242	M	19	40	(125)
Gear stage 1	243	M	19	41	(125)
Gear stage 2	244	M	19	42	(125)
Gear stage 3	245	M	19	43	(125)
Gear stage 4	246	M	19	44	(125)
Gear stage 5	247	M	19	45	(125)
Spindle speed	248	S	19	-1	(124)

Spindle-specific auxiliary functions, spindle 20					
System function	Index <n>	Type	Address extension	Value	Group
Spindle right	249	M	20	3	(126)
Spindle left	250	M	20	4	(126)
Spindle stop	251	M	20	5	(126)
Position spindle	252	M	20	19	(126)
Axis mode	253	M	20	70 [2)]	(126)
Automatic gear stage	254	M	20	40	(128)
Gear stage 1	255	M	20	41	(128)
Gear stage 2	256	M	20	42	(128)
Gear stage 3	257	M	20	43	(128)
Gear stage 4	258	M	20	44	(128)
Gear stage 5	259	M	20	45	(128)
Spindle speed	260	S	20	-1	(127)

Toolholder-specific auxiliary functions, T auxiliary functions					
System function	Index <n>	Type	Address extension	Value	Group
Tool selection	261	T	1	-1	129
Tool selection	262	T	2	-1	130
Tool selection	263	T	3	-1	131
Tool selection	264	T	4	-1	132
Tool selection	265	T	5	-1	133
Tool selection	266	T	6	-1	134
Tool selection	267	T	7	-1	135
Tool selection	268	T	8	-1	136
Tool selection	269	T	9	-1	137
Tool selection	270	T	10	-1	138
Tool selection	271	T	11	-1	139
Tool selection	272	T	12	-1	140
Tool selection	273	T	13	-1	141
Tool selection	274	T	14	-1	142
Tool selection	275	T	15	-1	143
Tool selection	276	T	16	-1	144
Tool selection	277	T	17	-1	145
Tool selection	278	T	18	-1	146
Tool selection	279	T	19	-1	147
Tool selection	280	T	20	-1	148

Toolholder-specific auxiliary functions, M6 auxiliary functions					
System function	Index <n>	Type	Address extension	Value	Group
Tool change	281	M	1	6 [1)]	149
Tool change	282	M	2	6 [1)]	150
Tool change	283	M	3	6 [1)]	151
Tool change	284	M	4	6 [1)]	152
Tool change	285	M	5	6 [1)]	153
Tool change	286	M	6	6 [1)]	154
Tool change	287	M	7	6 [1)]	155
Tool change	288	M	8	6 [1)]	156
Tool change	289	M	9	6 [1)]	157
Tool change	290	M	10	6 [1)]	158
Tool change	291	M	11	6 [1)]	159
Tool change	292	M	12	6 [1)]	160

Toolholder-specific auxiliary functions, M6 auxiliary functions					
System function	Index <n>	Type	Address extension	Value	Group
Tool change	293	M	13	6 1)	161
Tool change	294	M	14	6 1)	162
Tool change	295	M	15	6 1)	163
Tool change	296	M	16	6 1)	164
Tool change	297	M	17	6 1)	165
Tool change	298	M	18	6 1)	166
Tool change	299	M	19	6 1)	167
Tool change	300	M	20	6 1)	168

( ) The value can be changed.

1) The value is depends on the machine data:
MD22560 $MC_TOOL_CHANGE_M_MODE (M function for tool change)

2) The value can be preset with a different value using the following machine data:
MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR (M function for switching over to controlled axis mode (ext. mode))
MD20094 $MC_SPIND_RIGID_TAPPING_M_NR (M function for switching over to controlled axis mode)

Note
The value 70 is always output to the PLC.

3) The value is set using machine data:
MD22254 $MC_AUXFU_ASSOC_M0_VALUE (additional M function for program stop)

4) The value is set using machine data:
MD22256 $MC_AUXFU_ASSOC_M1_VALUE (additional M function for conditional stop)

5) The value is set using machine data:
MD10714 $MN_M_NO_FCT_EOP (M function for spindle active after reset)

6) The value is set using machine data:
MD26008 $MC_NIBBLE_PUNCH_CODE (definition of M functions)

7.2.2 Overview: Output behavior

Significance of the parameters listed in the following table:

Parameter	Meaning
Index <n>	Machine data index of the parameters of an auxiliary function
Output behavior	MD22080 $MC_AUXFU_PREDEF_SPEC[<n>], Bits 0 ... 18 Bits 19 ... 31: Reserved

Output behavior of the predefined auxiliary functions

System function	Index <n>																		
		Output behavior, bit																	
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Stop	0	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
Conditional stop	1	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
End of subroutine	2	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
	3	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
	4	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
Tool change	5	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Spindle right	6	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Spindle left	7	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Spindle stop	8	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Spindle positioning	9	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Axis mode	10	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
Automatic gear stage	11	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 1	12	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 2	13	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 3	14	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 4	15	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Gear stage 5	16	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
Spindle speed	17	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(1)	(0)	(0)	(0)	0	(0)	(1)
Feed	18	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(1)	(0)	0	(1)	0	(0)	(1)
Cutting edge selection	19	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)
DL	20	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)
Tool selection	21	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)
Stop (associated)	22	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
Conditional stop (associated)	23	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
End of subroutine	24	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
Nibbling	25	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	26	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	27	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)

System function	Index <n>																			
		Output behavior, bit																		
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Nibbling	28	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	29	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	30	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	31	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
Nibbling	32	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)

( ) The value can be changed.

Significance of the bits

Bit	Meaning
0	Acknowledgement "normal" after an OB1 cycle
	An auxiliary function with normal acknowledgment is output to the NC/PLC interface at the beginning of the OB1 cycle. The auxiliary function-specific change signal indicates to the PLC user program that the auxiliary function is valid. The auxiliary function is acknowledged as soon as organization block OB1 has run once. This corresponds to a complete PLC user cycle. The auxiliary function with normal acknowledgment is output in synchronism with the part program block in which it is programmed. If execution of the parts program block, e.g. path and/or positioning axis movements, is completed before acknowledgment of the auxiliary function, the block change is delayed until after acknowledgment by the PLC. In continuous-path mode, a constant path velocity can be maintained in conjunction with an auxiliary function with normal acknowledgment, if the auxiliary function is output by the PLC during the traversing motion and before reaching the end of the block.
1	Acknowledgement "quick" with OB40
	An auxiliary function with quick acknowledgment is output to the NC/PLC interface before the next OB1 cycle. The auxiliary function-specific change signal indicates to the PLC user program that the auxiliary function is valid. The auxiliary function is acknowledged immediately by the PLC basic program in the next OB40 cycle. Acknowledgment of the auxiliary function is not confirmation that the corresponding PLC user function has been executed. The auxiliary function is still executed in the OB1 cycle. Next output of the auxiliary functions to the PLC is therefore not possible until after this OB1 cycle has run completely. This is noticeable in continuous-path mode (drop in path velocity) especially if auxiliary functions with quick acknowledgment are output in several consecutive part program blocks. With auxiliary functions with quick acknowledgment, it cannot be guaranteed that the PLC user program will respond in synchronism with the block. Note Parameterization of the output behavior of auxiliary functions as "quick auxiliary functions" is only possible in conjunction with user-defined auxiliary functions.

Bit	Meaning
2	No predefined auxiliary function
	A predefined auxiliary function is treated like a user-defined auxiliary function with this setting. The auxiliary function then no longer triggers the corresponding system function but is only output to the PLC. Example: Reconfiguration of the "Position spindle" auxiliary function (index 9) to a user-defined auxiliary function with normal acknowledgment and output prior to the traversing motion. MD22080 $MC_AUXFU_PREDEF_SPEC [9] = 'H25' (100101B)
3	No output to the PLC
	The auxiliary function is not output to the PLC.
4	Spindle response after acknowledgement by the PLC
	The associated system function is only executed after acknowledgment by the PLC.
5	Output prior to motion
	The auxiliary function is output to the PLC before the traversing motions programmed in the part program block (path and/or block-related positioning axis movements).
6	Output during motion
	The auxiliary function is output to the PLC during the traversing motions programmed in the part program block (path and/or block-related positioning axis movements).
7	Output at block end
	The auxiliary function is output to the PLC after the traversing motions programmed in the part program block have been completed (path and/or block-related positioning axis movements).
8	Not output after block search, types 1, 2, 4
	Block search, types 1, 2, 4: The auxiliary function collected during the block search is not output.
9	Collection during block search with program test (type 5, SERUPRO)
	For a block search with program test, the auxiliary function is collected group-specific in the following system variables: • $AC_AUXFU_M_VALUE[<n>] • $AC_AUXFU_M_EXT[<n>] • $AC_AUXFU_M_STATE[<n>]
10	No output during block search with program test (type 5, SERUPRO)
	For block search with program test, the auxiliary function is not output to the PLC.
11	Cross-channel auxiliary function (SERUPRO)
	For block search with program test (SERUPRO), the help function is collected on a cross-channel basis in the global list of the auxiliary functions. Note For each auxiliary function group, only the last auxiliary function of the group is always collected.
12	Output performed via synchronized action (read only)
	The bit is set if the auxiliary function was output to the PLC via a synchronized action.
13	Implicit auxiliary function (read-only)
	The bit is set if the auxiliary function was implicitly output to the PLC.
14	Active M01 (read only)
	The bit is set if the auxiliary function, for active M01, was output to the PLC.

Bit	Meaning
15	No output during positioning test run
	During the run-in test, the auxiliary function is not output to the PLC.
16	Nibbling off
17	Nibbling on
18	Nibbling

Note

In the case of auxiliary functions for which no output behavior has been defined, the following default output behavior is active:

- Bit 0 = 1: Output duration one OB1 cycle
- Bit 7 = 1: Output at block end

7.2.3 Parameterization

7.2.3.1 Group assignment

The handling of the auxiliary functions for a block search is defined using the group assignment of an auxiliary function. The 168 auxiliary function groups available are subdivided into predefined and user-definable groups:

Predefined groups:	1 ... 4	10 ... 12	72 ... 168
User-defined groups:	5 ... 9	13 ... 71	

Each predefined auxiliary function is assigned, as standard, to an auxiliary function group. For most pre-defined auxiliary functions, this assignment can be changed using the following machine data:

MD22040 $MC_AUXFU_PREDEF_GROUP[<n>] (group assignment of predefined auxiliary functions)

If an auxiliary function is not assigned to any group, then a value of "0" should be entered into the machine data.

For the pre-defined auxiliary functions with the following indices <n>, the group assignment cannot be changed: 0, 1, 2, 3, 4, 22, 23, 24

Note

1. Auxiliary function group and block search

Auxiliary functions of the 1st auxiliary function group are, for a block search, only collected, but not output.

7.2.3.2 Type, address extension and value

An auxiliary function is programmed via the type, address extension and value parameters (see Section "Programming an auxiliary function (Page 439)").

Type

The identifier of an auxiliary function is defined via the "type," e.g.:

"M"	For additional function
"S"	For spindle function
"F"	For feed

The setting is made via the following machine data:

MD22050 $MC_AUXFU_PREDEF_TYPE[<n>] (type of predefined auxiliary functions)

Note

The "type" cannot be changed for predefined auxiliary functions.

Address extension

The "address extension" of an auxiliary function is for addressing different components of the same type. In the case of predefined auxiliary functions, the value of the "address extension" is the spindle number to which the auxiliary function applies.

The setting is made via the following machine data:

MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>] (address extension for predefined auxiliary functions)

Grouping together auxiliary functions

To assign an auxiliary function for all spindles of a channel to the same auxiliary function group, the value "-1" is entered for the "address extension" parameter.

Example:

The auxiliary function M3 (machine data index = 6) is assigned to the second auxiliary function group for all the channel's spindles.

```
MD22040 $MC_AUXFU_PREDEF_GROUP[ 6 ]        = 2
MD22050 $MC_AUXFU_PREDEF_TYPE[ 6 ]         = "M"
MD22060 $MC_AUXFU_PREDEF_EXTENSION[ 6 ]    = -1
MD22070 $MC_AUXFU_PREDEF_VALUE[ 6 ]        = 3
```

Value

The parameters "value" and "type" define the meaning of an auxiliary function, i.e. the system function that is activated on the basis of this auxiliary function.

The "value" of an auxiliary function is defined in the machine data:

MD22070 $MC_AUXFU_PREDEF_VALUE[<n>] (value of predefined auxiliary functions)

Note

The "value" cannot be changed for a predefined auxiliary function. For some predefined auxiliary functions, the "value" can be reconfigured via additional machine data (see Section "Associated auxiliary functions (Page 433)").

7.2.3.3 Output behavior

Parameter "Output behavior" defines when the predefined auxiliary function is output to the NC/PLC interface and when it is acknowledged by the PLC.

The setting is done via the following machine data:

MD22080 $MC_AUXFU_PREDEF_SPEC[<n>] (output behavior of predefined auxiliary functions)

Output behavior relative to motion

Output prior to motion

- The traversing motions (path and/or block-related positioning axis movements) of the previous part program block end with an exact stop.
- The auxiliary functions are output at the beginning of the current parts program block.
- The traversing motion of the actual part program block (path and/or positioning axis motion) is only started after acknowledgment of the auxiliary functions by the PLC:
  - Output duration one OB1 cycle (normal acknowledgment): after one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): after one OB40 cycle

Output during motion

- The auxiliary functions are output at the beginning of the traversing motions (path and/or positioning axis movements).
- The path velocity of the current parts program block is reduced so that the time to the end of the block is greater than the time to acknowledgment of the auxiliary functions by the PLC.
  - Output duration one OB1 cycle (normal acknowledgment): one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): one OB40 cycle

Output after motion

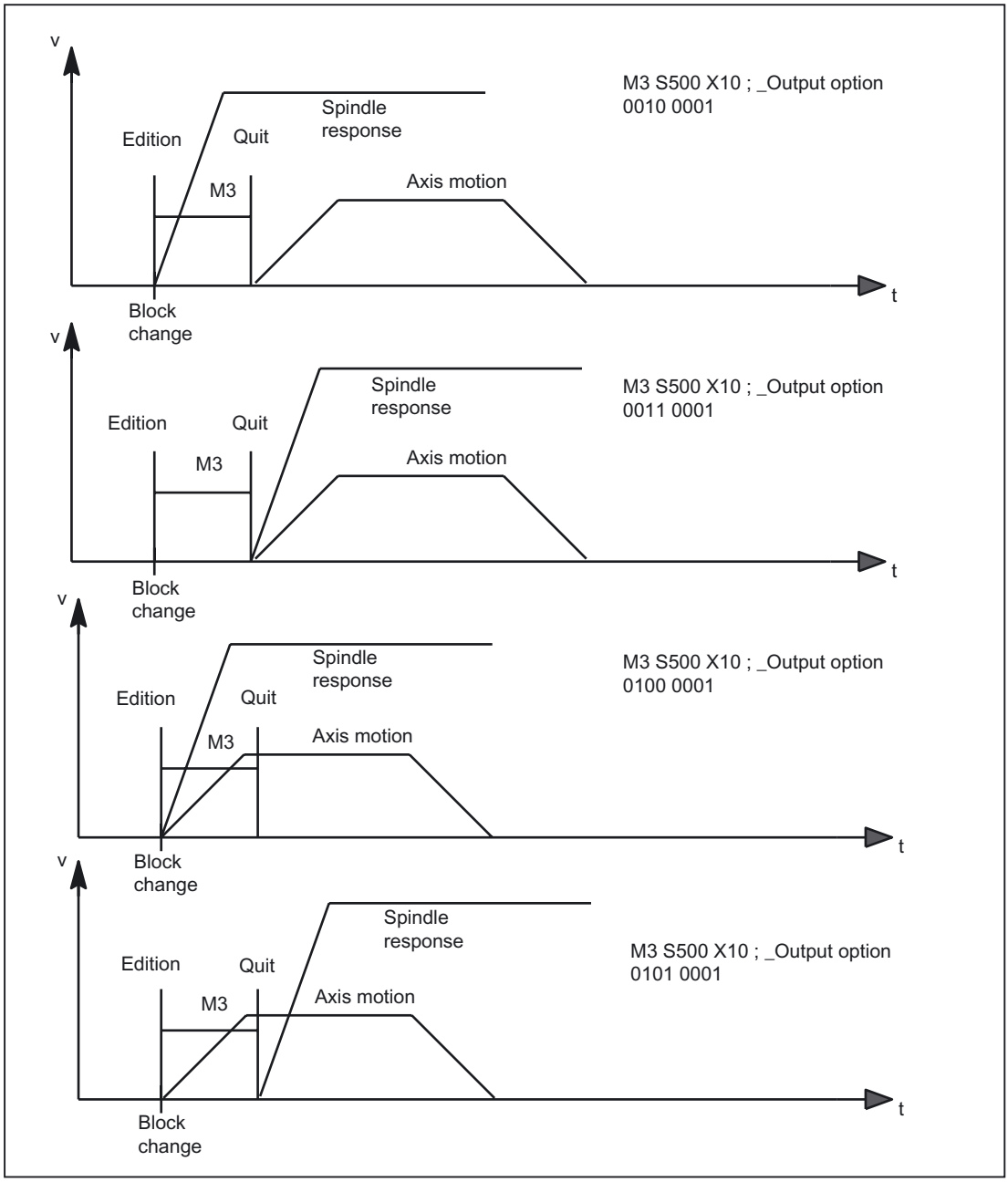
- The traversing motions (path and/or block-related positioning axis movements) of the current part program block end with an exact stop.
- The auxiliary functions are output after completion of the traversing motions.
- The block change is performed after acknowledgment of the auxiliary functions by the PLC:
  - Output duration one OB1 cycle (normal acknowledgment): after one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): after one OB40 cycle

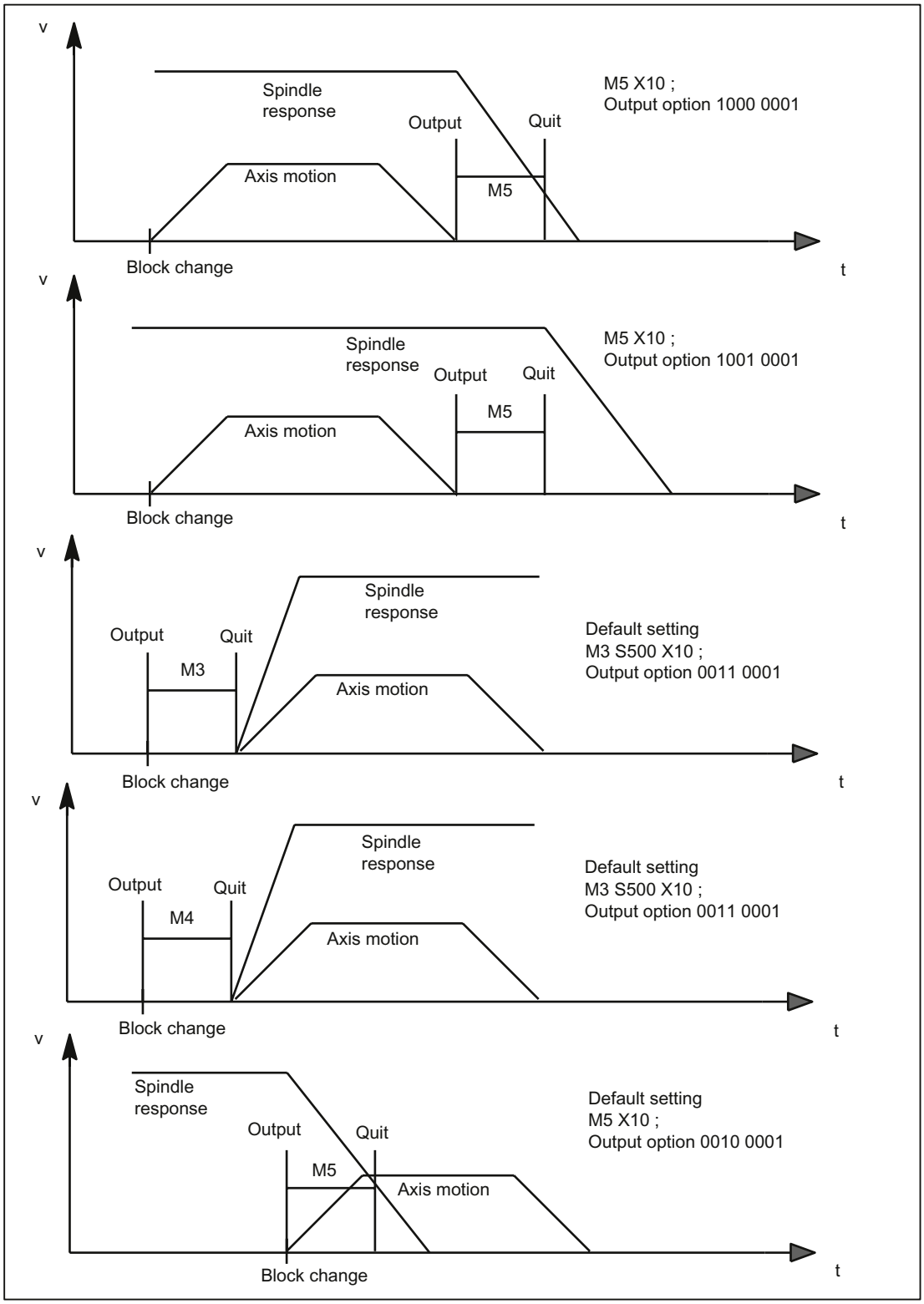
Examples of different output behavior

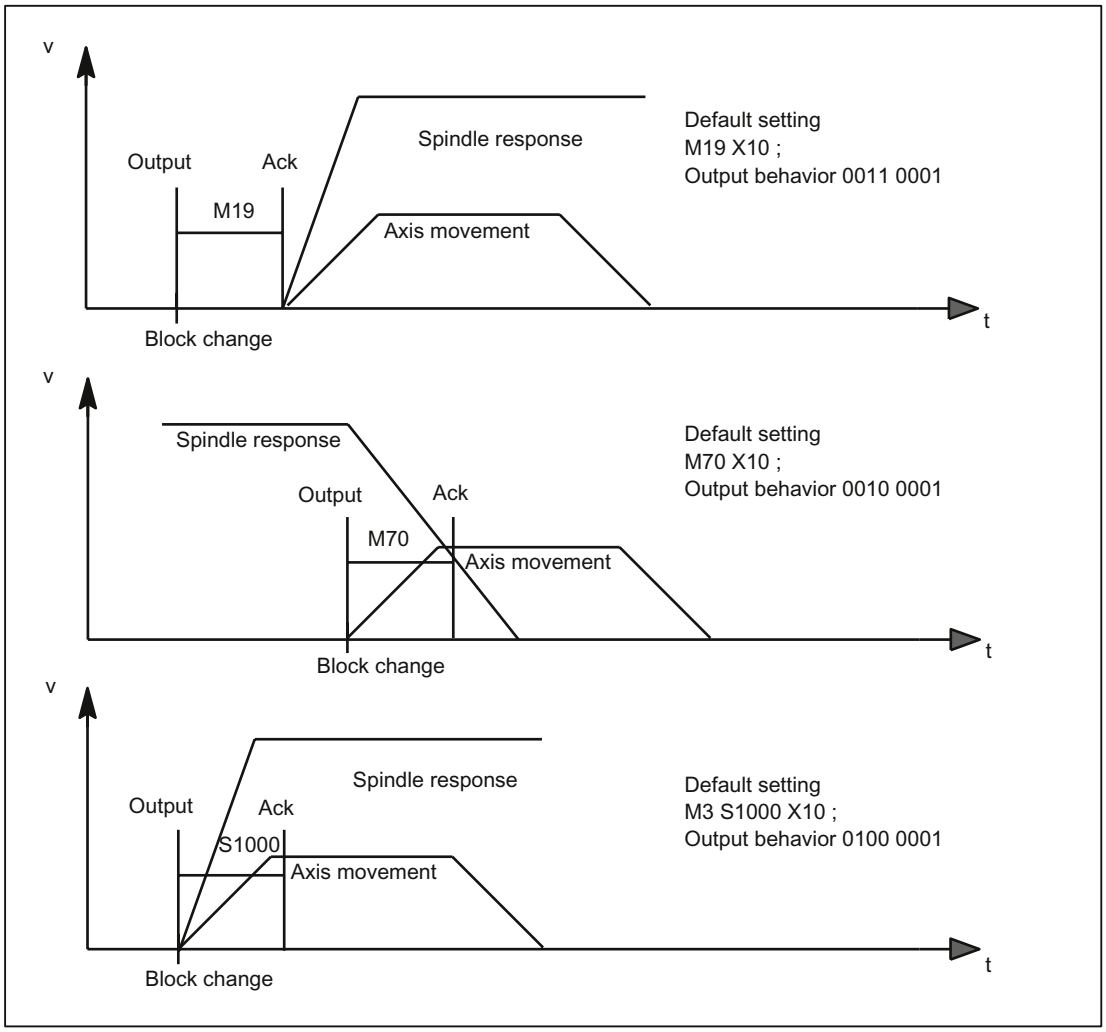
The following figures illustrate the differing behavior regarding:

- Output and acknowledgment of the auxiliary function
- Spindle response (speed change)
- Traverse movement (velocity change)

The binary values specified in the diagrams under "Output behavior" refer to the parameterized output behavior (MD22080).


v
M3 S500 X10 ; _Output option
0010 0001
Spindle
response
Edition
Quit
M3
Axis motion
t
Block
change
v
M3 S500 X10 ; _Output option
0011 0001
Spindle
response
Edition
Quit
M3
Axis motion
t
Block
change
v
M3 S500 X10 ; _Output option
0100 0001
Spindle
response
Edition
Quit
M3
Axis motion
t
Block
change
v
M3 S500 X10 ; _Output option
0101 0001
Spindle
response
Edition
Quit
M3
Axis motion
t
Block
change


v
Spindle response
Output
Quit
M5 X10 ;
Output option 1000 0001
Axis motion
M5
Block change
t
v
Spindle response
Output
Quit
M5 X10 ;
Output option 1001 0001
M5
Axis motion
Block change
t
v
Spindle response
Output
Quit
Default setting
M3 S500 X10 ;
Output option 0011 0001
M3
Axis motion
Block change
t
v
Spindle response
Output
Quit
Default setting
M3 S500 X10 ;
Output option 0011 0001
M4
Axis motion
Block change
t
v
Spindle response
Output
Quit
Default setting
M5 X10 ;
Output option 0010 0001
M5
Axis motion
Block change
t


v
Output
Ack
M19
Spindle response
Axis movement
Default setting
M19 X10 ;
Output behavior 0011 0001
Block change
t
v
Spindle response
Output
Ack
M70
Axis movement
Default setting
M70 X10 ;
Output behavior 0010 0001
Block change
t
v
Spindle response
Output
Ack
S1000
Axis movement
Default setting
M3 S1000 X10 ;
Output behavior 0100 0001
Block change
t

7.3 Userdefined auxiliary functions

There are two uses for user-defined auxiliary functions:

- Extension of predefined auxiliary functions
- User-specific auxiliary functions

Extension of predefined auxiliary functions

Because there is only one set of machine data for the predefined auxiliary functions, they can only ever be used to address one spindle of the channel. To address further spindles, user-defined auxiliary functions must be parameterized to supplement the predefined auxiliary functions.

Extension of predefined auxiliary functions refers to the "address extensions" parameter. The number of the spindle that the auxiliary function refers to is entered in the "address extension" parameter.

The relevant predefined auxiliary functions can be extended for the following system functions:

System function	Type		
		Address extension [1]	
			Value
Tool change	M	1	6
Spindle right	M	1	3
Spindle left	M	1	4
Spindle stop	M	1	5
Position spindle	M	1	19
Axis mode	M	1	70
Automatic gear stage	M	1	40
Gear stage 1	M	1	41
Gear stage 2	M	1	42
Gear stage 3	M	1	43
Gear stage 4	M	1	44
Gear stage 5	M	1	45
Spindle speed	S	1	-1
Tool selection	T	1	-1

[1] Address extension = 1 is the default value used in the auxiliary functions predefined in the machine data

Example:

Extension of the predefined auxiliary function for the system function "spindle right" for the second and third spindle of the channel.

Auxiliary function "spindle right" for the second spindle of the channel:

MD22010 $MC_AUXFU_ASSIGN_TYPE[n]	= "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[n]	**= 2**
MD22030 $MC_AUXFU_ ASSIGN_VALUE[n]	= 3

Auxiliary function "spindle right" for the third spindle of the channel:

MD22010 $MC_AUXFU_ ASSIGN_TYPE[m]	= "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[m]	**= 3**
MD22030 $MC_AUXFU_ ASSIGN_VALUE[m]	= 3

User-specific auxiliary functions

User-specific auxiliary functions have the following characteristics:

- User-specific auxiliary functions only activate user functions.
- No system functions can be activated by user-specific auxiliary functions.
- A user-specific auxiliary function is output to the PLC according to the parameterized output behavior.
- The functionality of a user-specific auxiliary function is implemented by the machine manufacturer/user in the PLC user program.

7.3.1 Parameterization

7.3.1.1 Maximum number of user-defined auxiliary functions

The maximum number of user-defined auxiliary function per channel can be parameterized via the machine data:

MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN (maximum number of user-defined auxiliary functions)

7.3.1.2 Group assignment

The handling of the auxiliary functions for a block search is defined using the group assignment of an auxiliary function. The 168 auxiliary function groups available are subdivided into predefined and user-definable groups:

Predefined groups:	1 ... 4		10 ... 12	72 ... 168
User-defined groups:		5 ... 9	13 ... 71	

Every user-defined auxiliary function is assigned as standard to the 1st auxiliary function group. The assignment can be changed using the following machine data:

MD22000 $MC_AUXFU_ASSIGN_GROUP[<n>] (group assignment of user-defined auxiliary functions)

If an auxiliary function is not assigned to any group, then a value of "0" should be entered into the machine data.

Note

1. Auxiliary function group and block search

Auxiliary functions of the 1st auxiliary function group are, for a block search, only collected, but not output.

7.3.1.3 Type, address extension and value

An auxiliary function is programmed via the type, address extension and value parameters (see Section "Programming an auxiliary function (Page 439)").

Type

The name of an auxiliary function is defined via the "type".

The identifiers for user-defined auxiliary functions are:

Type	Identifier	Meaning
"H"	Auxiliary function	User-specific auxiliary functions
"M"	Special function	Extension of predefined auxiliary functions
"S"	Spindle function	
"T"	Tool number	

The setting is made via the following machine data:

MD22010 $MC_AUXFU_ASSIGN_TYPE[<n>] (type of user-defined auxiliary functions)

Address extension

MD22020 $MC_AUXFU_ASSIGN_EXTENSION[<n>] (address extension user-defined auxiliary functions)

The functionality of the address extension is not defined in user-specific auxiliary functions. It is generally used to distinguish between auxiliary functions with the same "value".

Grouping together auxiliary functions

If all the auxiliary functions of the same type and value are assigned to the same auxiliary function group, a value of "-1" must be entered for the "address extension" parameter.

Example:

All user-specific auxiliary functions with the value "= 8" are assigned to the tenth auxiliary function group.

MD22000 $MC_AUXFU_ASSIGN_GROUP [1] = 10
MD22010 $MC_AUXFU_ ASSIGN_TYPE [1] = "H"
MD22020 $MC_AUXFU_ ASSIGN_EXTENSION [1] = -1
MD22030 $MC_AUXFU_ ASSIGN_VALUE [1] = 8

Value

MD22030 $MC_AUXFU_ASSIGN_VALUE[<n>] (value of user-defined auxiliary functions)

The functionality of the "value" parameter is not defined in user-specific auxiliary functions. The value is generally used to activate the corresponding PLC user function.

Grouping together auxiliary functions

If all the auxiliary functions of the same type and address extension are assigned to the same auxiliary function group, a value of "-1" must be entered for the "value" parameter.

Example:

All user-specific auxiliary functions with the address extension "= 2" are assigned to the eleventh auxiliary function group.

MD22000 $MC_AUXFU_ASSIGN_GROUP [2]	= 11
MD22010 $MC_AUXFU_ ASSIGN_TYPE [2]	= "H"
MD22020 $MC_AUXFU_ ASSIGN_EXTENSION [2]	= 2
MD22030 $MC_AUXFU_ ASSIGN_VALUE [2]	= -1

7.3.1.4 Output behavior

The "output behavior" of user-defined auxiliary functions can be parameterized via the machine data:

MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>] (output behavior of user-defined auxiliary functions)

For a description of the individual output parameters, see the "Output behavior (Page 423)" section of the predefined auxiliary functions. The information given there can be applied analogously to the output behavior of user-defined auxiliary functions.

7.4 Associated auxiliary functions

Function

Associated auxiliary functions are user-defined auxiliary functions that have the same effect as the corresponding predefined auxiliary functions. User-defined auxiliary functions can be associated for the following predefined auxiliary functions:

- M0 (stop)
- M1 (conditional stop)

Requirements

The precondition for association of a user-defined auxiliary function with one of the predefined auxiliary functions mentioned is parameterization of a user-defined auxiliary function. Only "M" is allowed as a "type" parameter of the user-defined auxiliary function.

Parameter assignment

Association of a user-defined auxiliary function with one of the predefined auxiliary functions mentioned is set in the machine data:

MD22254 $MC_AUXFU_ASSOC_M0_VALUE (additional M function for program stop)

MD22256 $MC_AUXFU_ASSOC_M1_VALUE (additional M function for conditional stop)

Group assignment

The group assignment of an associated user-defined auxiliary function is always the group assignment of the corresponding predefined auxiliary function.

Application

Associated auxiliary functions can be used in:

- Main program
- Subroutine
- Cycle

Note

Associated auxiliary functions may not be used in synchronized actions.

NC/PLC interface signals

In the case of an associated user-defined auxiliary function, the same signals are output to the NC/PLC interface as for the corresponding predefined auxiliary function. To distinguish which auxiliary function has actually been programmed, the value of the user-defined auxiliary function ("value" parameter) is output as the value of the auxiliary function. This means it is possible to distinguish between predefined and user-defined auxiliary functions in the PLC user program.

Note

A change in machine data MD22254 and/or MD22256 may require corresponding adjustment of the PLC user program:

Specific NC/PLC interface signals

The following specific NC/PLC interface signals are available:

- DB21, ... DBX318.5 (associated M00/M01 active) feedback signal
- DB21, ... DBX30.5 (activate associated M01) activation signal

Boundary conditions

Please note the following boundary conditions:

- A user-defined auxiliary function may not be multiply associated.
- Predefined auxiliary functions (e.g. M3, M4, M5 etc.) may not be associated.

Example

Associating the user-defined auxiliary function M123 with M0:

MD22254 $MC_AUXFU_ASSOC_M0_VALUE = 123

The user-defined auxiliary function M123 thus has the same functionality as M0.

7.5 Type-specific output behavior

Function

The output behavior of auxiliary functions relative to a traversing motions programmed in the parts program block can be defined type-specifically.

Parameter assignment

Parameters are assigned to type-specific output behavior via the machine data:

MD22200 $MC_AUXFU_**M**_SYNC_TYPE (output time for M functions)

MD22210 $MC_AUXFU_**S**_SYNC_TYPE (output time for S functions)

MD22220 $MC_AUXFU_**T**_SYNC_TYPE (output time for T functions)

MD22230 $MC_AUXFU_**H**_SYNC_TYPE (output time for H functions)

MD22240 $MC_AUXFU_**F**_SYNC_TYPE (output time for F functions)

MD22250 $MC_AUXFU_**D**_SYNC_TYPE (output time for D functions)

MD22252 $MC_AUXFU_**DL**_SYNC_TYPE (output time for DL functions)

The following output behaviors can be parameterized:

MD $MC_AUXFU_**xx**_SYNC_TYPE = <value>

Value	Output behavior
0	Output prior to motion
1	Output during motion
2	Output at block end
3	No output to the PLC
4	Output according to the output behavior defined with MD22080

For a description of the various output behaviors, see the section titled "Output behavior (Page 423)".

Note

For the output behavior that can be set for each type of auxiliary function, please refer to the "Detailed Description of Machine Data" Parameter Manual.

Example

Output of auxiliary functions with different output behaviors in a part program block with traverse movement.

Output behavior for which parameters have been assigned:

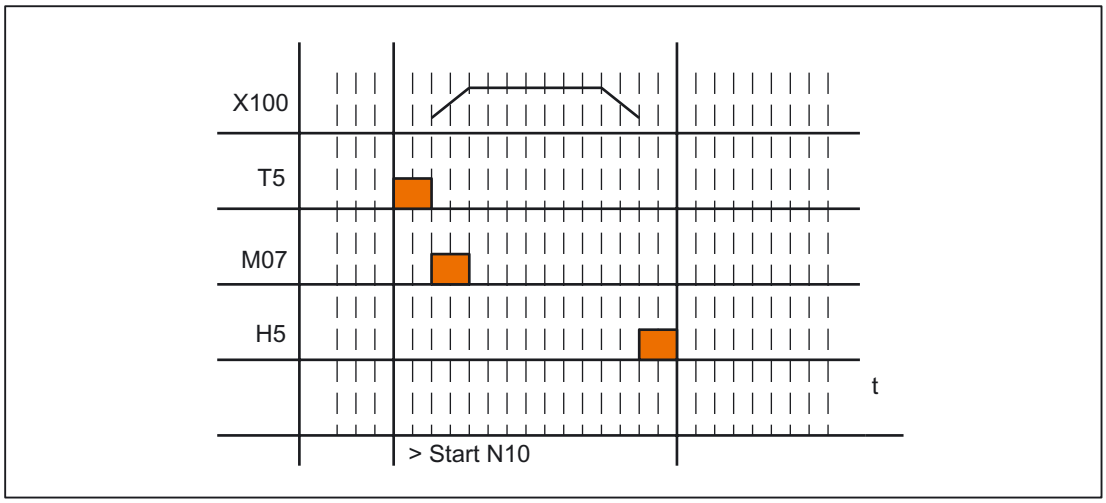
MD22200 $MC_AUXFU_M_SYNC_TYPE = 1	⇒	M function: Output **during** motion
MD22220 $MC_AUXFU_T_SYNC_TYPE = 0	⇒	T function: Output **prior to** motion
MD22230 $MC_AUXFU_H_SYNC_TYPE = 2	⇒	H function: Output **at the end of the block**

Parts program block:

Program code

```
...
N10 G01 X100 M07 H5 T5
...
```

Time sequence for auxiliary function output:



7.6 Priorities of the output behavior for which parameters have been assigned

The following priorities must be observed for the following areas in connection with the parameterized output behavior of an auxiliary function:

- Output duration (normal / quick acknowledgement)
- Output relative to motion (prior to / during / after the motion)

As a general rule, the parameterized output behavior with lower priority becomes active if no output behavior with higher priority has been parameterized.

Output duration

The following priorities apply to the output duration:

Priority	Output behavior	Defined via:
Highest	Auxiliary function-specific	Part program instruction: QU(…) (see Section "Programmable output duration (Page 441)")
↓	Auxiliary function-specific	MD22035 $MC_AUXFU_ASSIGN_SYNC[<n>] MD22080 $MC_AUXFU_PREDEF_SYNC[<n>]
↓	Group-specific	MD11110 $MC_AUXFU_GROUP_SPEC[<n>]
Lowest	Not defined	Default output behavior: Output duration one OB1 cycle

Output relative to motion

The following rules apply to output relative to motion:

Priority	Output behavior	Defined via:
Highest	Auxiliary function-specific	MD22035 $MC_AUXFU_ASSIGN_SYNC[<n>] MD22080 $MC_AUXFU_PREDEF_SYNC[<n>]
↓	Group-specific	MD11110 $MC_AUXFU_GROUP_SPEC[<n>]
↓	Type-specific	MD22200 $MC_AUXFU_M_SYNC_TYPE MD22210 $MC_AUXFU_S_SYNC_TYPE MD22220 $MC_AUXFU_T_SYNC_TYPE MD22230 $MC_AUXFU_H_SYNC_TYPE MD22240 $MC_AUXFU_F_SYNC_TYPE MD22250 $MC_AUXFU_D_SYNC_TYPE MD22252 $MC_AUXFU_DL_SYNC_TYPE
Lowest	Not defined	Default output behavior: Output at block end

Note

Part program blocks without path motion

In a part program block without a path motion (even those with positioning axes and spindles), the auxiliary functions are all output immediately in a block.

7.7 Programming an auxiliary function

Syntax

An auxiliary function is programmed in a part program block with the following syntax:
`<Type>[<Address extension>=]<Value>`

Note

If no address extension is programmed, the address extension is implicitly set = 0.

Predefined auxiliary functions with the address extension = 0 always refer to the master spindle of the channel.

Symbolic addressing

The values for the "address extension" and "value" parameters can also be specified symbolically. The symbolic name for the address extension must then be stated in brackets.

Example:

Symbolic programming of the auxiliary function M3 (spindle right) for the first spindle:

Program code	Comment
`DEF SPINDEL_NR=1`	`; First spindle in the channel`
`DEF DREHRICHTUNG=3`	`; Clockwise direction of rotation`
`N100 M[SPINDEL_NR] = DREHRICHTUNG`	`; corresponding to: M1=3`

Note

If you use symbolic names to program an auxiliary function, the symbolic name is not transferred when the auxiliary function is output to the PLC. The corresponding numerical value is transferred instead.

Examples

Example 1: Programming of predefined auxiliary functions

```
Program code          Comment
N10 M3              ; "Spindle right" for the master spindle of the channel.
N20 M0=3            ; "Spindle right" for the master spindle of the channel.
N30 M1=3            ; "Spindle right" for the 1st spindle of the channel.
N40 M2=3            ; "Spindle right" for the 2nd spindle of the channel.
```

Example 2: Programming examples of auxiliary functions with the corresponding values for output to the PLC

```
Program code                          Comment
DEF Coolant=12                      ; Output to PLC: - - -
DEF Lubricant=130                   ; Output to PLC: - - -
H[coolant]=lubricant                ; Output to PLC: H12=130
H=coolant                           ; Output to PLC: H0=12
H5                                  ; Output to PLC: H0=5
H=5.379                             ; Output to PLC: H0=5.379
H17=3.5                             ; Output to PLC: H17=3.5
H[coolant]=13.8                     ; Output to PLC: H12=13.8
H='HFF13'                           ; Output to PLC: H0=65299
H='B1110'                           ; Output to PLC: H0=14
H5.3=21                             ; Error
```

7.8 Programmable output duration

Function

User-specific auxiliary functions, for which the output behavior "Output duration of an OB1 cycle (slow acknowledgement)" was parameterized, can be defined for individual outputs via the parts program guide `QU` (Quick) for auxiliary functions with quick acknowledgement.

Syntax

An auxiliary function with quick acknowledgment is defined in a part program block with the following syntax:

```
<Type>[<Address extension>]=QU(<Value>)
```

Example

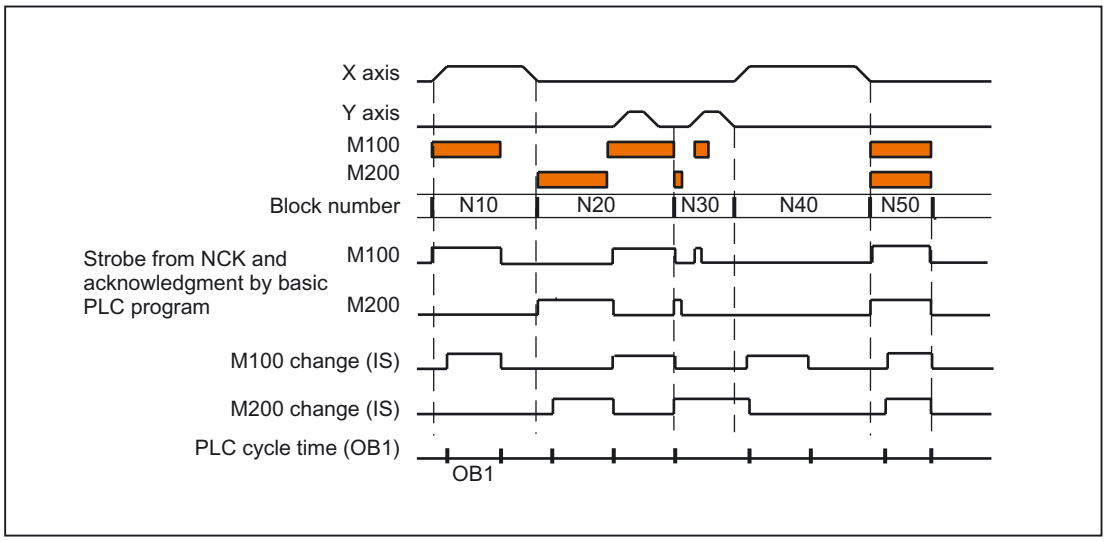
Different behavior for the output of the auxiliary functions M100 and M200 in a parts program. The output behavior of the auxiliary functions is parameterized as follows:

- M100
  - Output duration one OB1 cycle (slow acknowledgment)
  - Output during motion
- M200
  - Output duration one OB1 cycle (slow acknowledgment)
  - Output prior to motion

Program code	Comment
N10 G94 G01 X50 M100	; Output of M100: during the motion
	; Acknowledgment: slow
N20 Y5 M100 M200	; Output of M200: prior to the motion
	; Output of M100: during the motion
	; Acknowledgment: slow
N30 Y0 M=QU(100) M=QU(200)	; Output of M200: prior to the motion
	; Output of M100: during the motion
	; Acknowledgement: quick
N40 X0	
N50 M100 M200	; Output of M200: immediate 1)
	; Output of M100: immediate 1)
	; Acknowledgment: slow
M17	

1) Without a traverse movement, auxiliary functions are always output to the PLC immediately.

The following figure shows the time sequence of the part program. Please note the time difference during the processing of parts program blocks N20 and N30.



7.9 Auxiliary function output to the PLC

Function

On output of an auxiliary function to the PLC, the following signals and values are transferred to the NC/PLC interface:

- Change signals
- "Address extension" parameter
- "Value" parameter

Data areas in the NC/PLC interface

The change signals and values of the auxiliary functions are within the following data areas in the NC/PLC interface:

- Change signals for auxiliary function transfer from NC channel:

  DB21, ... DBB58 - DBB67
- Transferred M and S functions:

  DB21, ... DBB68 - DBB112
- Transferred T, D and DL functions:

  DB21, ... DBB116 - DBB136
- Transferred H and F functions:

  DB21, ... DBB140 - DBB190
- Decoded M signals (M0 - M99):

  DB21, ... DBB194 - DBB206 (dynamic M functions)

For information on the access procedure to the NC/PLC interface, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)".

A detailed description of the above data areas in the NC/PLC interface can be found in:

References:
List Manual, Lists, Book 2; PLC User Interfaces,
Section: Channel-specific signals (DB 21 - DB 30)"

7.10 Auxiliary functions without block change delay

Function

For auxiliary functions with parameterized and/or programmed output behavior, too:

- "Output duration one OB40 cycle (quick acknowledgment)"
- "Output before the motion" or "Output during the motion"

there may be drops in velocity in continuos-path mode (short traverse paths and high velocities). This the system has to wait for acknowledgment of the auxiliary function by the PLC toward the end of the block. To avoid these velocity drops, the block change can be made irrespective of whether such auxiliary functions have been acknowledged:

Parameter assignment

Suppression of the block change delay with quick auxiliary functions is set via the machine data:

MD22100 $MC_AUXFU_QUICK_BLOCKCHANGE (block change delay with quick auxiliary functions)

Value	Meaning
0	In the case of quick auxiliary function output to the PLC, the block change is delayed until acknowledgment by the PLC (OB40).
1	In the case of quick auxiliary function output to the PLC, the block change is not delayed.

Boundary conditions

Synchronism of auxiliary functions that are output without a block change delay is no longer ensured for the part program block in which they are programmed. In the worst case scenario, acknowledgment comes one OB40 cycle and execution of the auxiliary function comes one OB1 cycle after the change to the next part program block.

7.11 M function with an implicit preprocessing stop

Function

Triggering a preprocessing stop in conjunction with an auxiliary function can be programmed explicitly via the STOPRE part program command. Always triggering a preprocessing stop in M function programming can be parameterized for each M function via the following machine data:

MD10713 $MN_M_NO_FCT_STOPRE[<n>] (M function with preprocessing stop)

Example

The user-defined M function M88 is intended to trigger a preprocessing stop.

Parameter assignment:

MD10713 $MN_M_NO_FCT_STOPRE [0] = 88

Application:

Parts program (extract)

```
Program code              Comment
...
N100 G0 X10 M88           ; Traversing motion and implicit preprocessing stop via M88
N110 Y=R1                 ; N110 is only interpreted after the traversing motion has
                            been completed and the M function has been acknowledged.
...
```

Boundary conditions

If a subroutine called indirectly via an M function in a part program in one of the following ways, no preprocessing stop is performed:

- MD10715 $MN_M_NO_FCT_CYCLE (M function to be replaced by subroutine)
- M98 (ISO dialect T / ISO dialect M)

7.12 Response to overstore

Overstore

On the SINUMERIK operator interface, before starting the following functions:

- NC START of a part program
- NC START to resume an interrupted part program

the auxiliary functions that are output at the start can be changed by the "Overstore" function.

Possible applications include:

- Addition of auxiliary functions after block search
- Restoring the initial state to position a part program

Types of auxiliary functions that can be overstored

The following types of auxiliary functions can be overstored:

- M (special function)
- S (spindle speed)
- T (tool number)
- H (aux. function)
- D (tool offset number)
- DL (additive tool offset)
- F (feed)

Duration of validity

An overstored auxiliary function, e.g. M3 (spindle right), is valid until it is overwritten by another auxiliary function from the same auxiliary function group, by additional overstoring or by programming in a part program block.

7.13 Behavior during block search

7.13.1 Auxiliary function output during type 1, 2, and 4 block searches

Output behavior

In the case of type 1, 2, and 4 block searches, the auxiliary functions are collected on the basis of specific groups. The last auxiliary function in each auxiliary function group is output after NC-START in a separate part program block before the actual reentry block, and has the following output behavior:

- Output duration of one OB1 cycle (normal acknowledgement)
- Output prior to motion

Output control

Whether or not the auxiliary function is output to the PLC after a block search can be configured via bit 8 of the machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]

  (output behavior of predefined auxiliary functions)

  where <n> = system function index (0 ... 32)
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]

  (output behavior of user-defined auxiliary functions)

  where <n> = auxiliary function index (0 ... 254)
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]

  (output behavior of the auxiliary functions in a group)

  where <n> = group index (0 ... 63)

Bit	Value	Meaning
10	0	Output during type 1, 2, and 4 block searches
	1	No output during type 1, 2, and 4 block searches

This behavior does not affect the display and does not affect variables $AC_AUXFU_STATE[<n>], $AC_AUXFU_VALUE[<n>], and $AC_AUXFU_EXT[<n>].
The auxiliary functions are always regarded as collected after a block search, even though they are not output to the PLC.

During collection, an auxiliary function that is not output after a block search also overwrites an auxiliary function whose bit 8 is not set.

The user can scan the collected auxiliary functions after a block search and, under certain circumstances, output them again by means of the subprogram or synchronized actions.

Note

The following auxiliary functions are not collected:

- Auxiliary functions which are not assigned to any auxiliary function group.
- Auxiliary functions which are assigned to the first auxiliary function group.

Overstorage of auxiliary functions

After completion of a block search, the collected auxiliary functions are ouput on the next NC-START. If it is necessary to output additional auxiliary functions, they can be added via the "Overstore" function (see Section "Response to overstore (Page 446)").

M19 behavior (position spindle)

After a block search, the last spindle positioning command programmed with M19 is always carried out, even if other spindle-specific auxiliary functions are programmed between the part program with M19 and the target block. Setting the necessary spindle enables must therefore be derived from the interface signals of the traverse commands in the PLC user program:

DB31, ... DBX64.6/64.7 (traversing command minus/plus)

In this case, the spindle-specific auxiliary functions M3, M4, and M5 are not suitable because they might not be output to the PLC until after the spindle positioning.

For detailed information on the block search, see Section "K1: Mode group, channel, program operation, reset response (Page 489)".

7.13.2 Assignment of an auxiliary function to a number of groups

Function

User-defined auxiliary functions can also be assigned to multiple groups via the group assignment (MD22000 $MC_AUXFU_ASSIGN_GROUP). During the block search these auxiliary functions are collected for all the configured groups.

Note

Predefined auxiliary functions can only be assigned to one group.

Example

The DIN includes the following M-commands for coolant output:

- M7: Coolant 2 ON
- M8: Coolant 1 ON
- M9: Coolants 1 and 2 OFF

Consequently, both coolants can also be active together:

- If M7 and M8 are collected in two separate groups (e.g. groups 5 and 6)
- If M9 has to be assigned to these two groups, e.g.
  - Group 5: M7, M9
  - Group 6: M8, M9

Parameterization:

MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN = 4

MD22000 $MC_AUXFU_ASSIGN_GROUP [0] = 5

MD22000 $MC_AUXFU_ASSIGN_GROUP [1] = 5

MD22000 $MC_AUXFU_ASSIGN_GROUP [2] = 6

MD22000 $MC_AUXFU_ASSIGN_GROUP [3] = 6

MD22010 $MC_AUXFU_ASSIGN_TYPE [0] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [1] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [2] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [3] = M

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [0] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [1] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [2] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [3] = 0

MD22030 $MC_AUXFU_ASSIGN_VALUE [0] = 7

MD22030 $MC_AUXFU_ASSIGN_VALUE [1] = 9

MD22030 $MC_AUXFU_ASSIGN_VALUE [2] = 8

MD22030 $MC_AUXFU_ASSIGN_VALUE [3] = 9

MD22035 $MC_AUXFU_ASSIGN_SPEC [0] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [1] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [2] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [3] = 'H121'

Part program (section):

```
Program code
...
N10 ... M8
N20 ... M9
N30 ... M7
...
```

During the block search, the auxiliary function M9 is collected for groups 5 and 6.

Scan of the collected M auxiliary functions:

M function of the fifth group: $AC_AUXFU_M_VALUE [4] = 7

M function of the sixth group: $AC_AUXFU_M_VALUE [5] = 9

7.13.3 Time stamp of the active M auxiliary function

When outputting collected auxiliary functions following a block search, attention must be paid to the sequence during collecting. For this reason, each group is assigned a time stamp which can be queried on a group-specific basis by way of the system variable below:

$AC_AUXFU_M_TICK[<n>] (time stamp of the active M auxiliary function)

7.13.4 Determining the output sequence

Function

The following predefined procedure is provided to simplify the process of determining the output sequence of M auxiliary functions for the programmer:

```
AUXFUMSEQ(VAR INT _NUM_IN, VAR INT _M_IN[], VAR INT _EXT_IN[], VAR
INT _NUM_OUT, VAR INT _M_OUT[], VAR INT _EXT_OUT[])
```

Input parameters:

`VAR INT _NUM_IN`:	Number of relevant M commands
`VAR INT _M_IN[]`:	Field of relevant M codes
`VAR INT _EXT_IN[]`:	Field of relevant M address extensions

Output parameters:

`VAR INT _NUM_OUT`:	Number of determined M codes
`VAR INT _M_OUT[]`:	Field of determined M codes
`VAR INT _EXT_OUT[]`:	Field of determined M address extensions

The function determines the sequence in which the M auxiliary functions, which have been collected on a group-specific basis, are output for the predefined M codes. The sequence is determined from the collection times $AC_AUXFU_M_TICK[<n>] (see Section "Time stamp of the active M auxiliary function (Page 451)").

A particular M code is only taken into account once, even if it belongs to more than one group. If the number of relevant M commands is less than or equal to 0, all the collected M codes are output. The number of relevant M commands is limited to 64.

Example

M commands for coolant output:

- M7: Coolant 2 ON
- M8: Coolant 1 ON
- M9: Coolants 1 and 2 OFF

Group assignment:

- Group 5: M7, M9
- Group 6: M8, M9

Part program (section):

Program code

```
...
N10 ... M8
N20 ... M9
N30 ... M7
...
```

During block searches, the auxiliary functions are collected on the basis of specific groups. The last auxiliary function in an auxiliary function group is output to the PLC following a block search:

- Group 5: M7
- Group 6: M9

If they are output in the sequence M7 → M9, no coolant is then active. However, coolant 2 would be active during the execution of the program. Therefore, the correct output sequence for the M auxiliary functions is determined with an ASUP which contains the predefined procedure `AUXFUMSEQ(…)`:

Program code

```
DEF INT _I, _M_IN[3], _EXT_IN[3], _NUM_OUT, _M_OUT[2], _EXT_OUT[2]
_M_IN[0]=7 _EXT_IN[0]=0
_M_IN[1]=8 _EXT_IN[1]=0
_M_IN[2]=9 _EXT_IN[2]=0
AUXFUMSEQ(3,_M_IN,_EXT_IN,_NUM_OUT,_M_OUT,_EXT_OUT)
FOR _I = 0 TO _NUM_OUT-1
    M[_EXT_OUT[_I]]=_M_OUT[_I]
ENDFOR
```

7.13.5 Output suppression of spindle-specific auxiliary functions

Function

In certain situations, such as a tool change, it may be necessary not to output the spindle-specific auxiliary functions collected during the block search in action blocks, but to delay output, for example, until after a tool change. The automatic output of the spindle-specific auxiliary functions after a block search may be suppressed for this purpose. Output can then be performed manually later by overstoring or by an ASUB.

Parameterization

Suppression of the automatic output of the spindle-specific auxiliary functions after a block search is set via machine data:

MD11450 $MN_SEARCH_RUN_MODE (behavior after a block search)

Bit	Value	Meaning
2	0	The output of the spindle-specific auxiliary functions is performed in the action blocks.
	1	Output of the auxiliary functions is suppressed in the action blocks.

System variables

The spindle-specific auxiliary functions are always stored in the following system variables during block searches, irrespective of the parameter assignment described above:

System variable	Description	
$P_SEARCH_S [<n>]	Accumulated spindle speed	
	Range of values:	0 ... Smax
$P_SEARCH_SDIR [<n>]	Accumulated spindle direction of rotation	
	Range of values:	3, 4, 5, -5, -19, 70
$P_SEARCH_SGEAR [<n>]	Accumulated spindle gear stage M function	
	Range of values:	40 ... 45
$P_SEARCH_SPOS [<n>]	Accumulated spindle position	
	Range of values:	0 ... MD30330 $MA_MODULO_RANGE (size of the module range)
	or	
	Accumulated traversing path	
	Range of values:	-100.000.000 ... 100.000.000
$P_SEARCH_SPOSMODE [<n>]	Accumulated position approach mode	
	Range of values:	0 ... 5

For later output of the spindle-specific auxiliary functions, the system variables can be read in an ASUB, for example, and output after the action blocks are output:

DB21, ... DBX32.6 = 1 (last action block active)

Note

The contents of the system variables $P_S, $P_DIR and $P_SGEAR may be lost after block search due to synchronization operations.

For more detailed information on ASUB, block search and action blocks, see Section "K1: Mode group, channel, program operation, reset response (Page 489)".

Example

Block search for contour with suppression of output of the spindle-specific auxiliary functions and start of an ASUB after output of action blocks.

Parameterization: MD11450 $MN_SEARCH_RUN_MODE, bit 2 = 1

After the block search on N55, the ASUB is started.

Part program:

```
Program code              Comment
N05 M3 S200               ; Spindle 1
N10 G4 F3
N15 SPOS=111              ; Spindle 1 is positioned to 111 degrees in the ASUB
N20 M2=4 S2=300           ; Spindle 2
N25 G4 F3
N30 SPOS[2]=IC(77)        ; Spindle 2 traverses incrementally by 77 degrees
N55 X10 G0                ; Target block
N60 G4 F10
N99 M30
```

ASUB:

```
Program code                                    Comment
PROC ASUP_SAVE
MSG ("Output of the spindle functions")
DEF INT SNR=1
AUSG_SPI:
M[SNR]=$P_SEARCH_SGEAR[SNR]                     ; Output gear stage
S[SNR]=$P_SEARCH_S[SNR]                         ; Output speed (for M40, a suitable
                                                  gear stage is determined)
M[SNR]=$P_SEARCH_SDIR[SNR]                      ; Output direction of rotation,
                                                  positioning or axis mode
SNR=SNR+1                                       ; Next spindle
REPEAT AUSG_SPI P=$P_NUM_SPINDLES-1             ; For all spindles
MSG("")
REPOSA
RET
```

Explanation of example

If the number of spindles is known, outputs of the same type can be written in one part program block to reduce program runtime.

Output of $P_SEARCH_SDIR should be made in a separate part program block because spindle positioning or switchover to axis mode in conjunction with the gear change can cause an alarm.

If the ASUB which has been started is ended with `REPOSA`, spindle 1 remains at position 111 degrees, while spindle 2 is repositioned at position 77 degrees.

If a different response is required, the program sequence for block search (for example) "`N05 M3 S...`" and "`N30 SPOS[2] = IC(...)`" requires special treatment.

Whether block search is active can be ascertained in the ASUB via the system variable $P_SEARCH.

```
$P_SEARCH==1 ; Block search active
```

In the case of incremental positioning after speed control operation, the path to be traversed is defined but, in some cases, the final position reached only becomes known during positioning. This is the case, for example, during position calibration on crossing the zero mark when switching on position control. For this reason, the distance programmed after the zero position is accepted as the REPOS position (`REPOSA` in the ASUB).

Supplementary conditions

Collected S values

The meaning of an S value in the part program depends on the feed type that is currently active:

G93, G94, G95, G97, G971:	The S value is interpreted as the speed
G96, G961:	The S value is interpreted as a constant cutting rate

If the feed operation is changed (e.g. for a tool change) before output of the system variable $P_SEARCH_S, the original setting from the target block in the part program must be restored to avoid use of the wrong type of feed.

Collected direction of rotation

For output of the direction of rotation, the system variable $P_SEARCH_SDIR is assigned default value "-5" at the start of the block search. This value has no effect on output.

This ensures that the last spindle operating mode is retained for a block search across program sections in which spindles are not programmed with a direction of rotation, positioning or axis mode.

The programming of M19, SPOS, and SPOSA is collected as "M-19" (internal M19) in the system variables $P_SEARCH_SDIR as an alternative to M3, M4, M5, and M70.

For the output of "M-19", the positioning data is read internally from the system variables $P_SEARCH_SPOS and $P_SEARCH_SPOSMODE. Both system variables can also be written to, in order, for example, to make corrections.

Note

Because of the assignments described above (e.g. M[<n>] = $P_SEARCH_SDIR[<n>]), the values "–5" and "–19" generally remain hidden from the user and only have to be observed in the case of special evaluation of the system variables in the ASUB.

7.13.6 Auxiliary function output with a type 5 block search (SERUPRO)

Output behavior

In the case of type 5 block searches (SERUPRO), an auxiliary function can be output to the PLC during the block search and/or collected on a group-specific basis in the following system variables:

- $AC_AUXFU_PREDEF_INDEX[<n>] (index of a predefined auxiliary function)
- $AC_AUXFU_TYPE[<n>] (type of auxiliary function)
- $AC_AUXFU_STATE[<n>] (output state of the auxiliary function)
- $AC_AUXFU_EXT[<n>] (address extension of the auxiliary function)
- $AC_AUXFU_VALUE[<n>] (value of the auxiliary function)

For a description of the system variables, see Section "Querying system variables (Page 473)".

Output control

Whether an auxiliary function is output to the PLC during a type 5 block search (SERUPRO) and/or collected on a group-specific basis in the following system variables can be configured via bits 9 and 10 of the machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]

  (output behavior of predefined auxiliary functions)

  where <n> = system function index (0 ... 32)
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]

  (output behavior of user-defined auxiliary functions)

  where <n> = auxiliary function index (0 ... 254)
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]

  (output behavior of the auxiliary functions in a group)

  where <n> = group index (0 ... 63)

Bit	Value	Meaning
9	0	No collection during type 5 block searches (SERUPRO)
	1	Collection during type 5 block searches (SERUPRO)
10	0	Output during type 5 block searches (SERUPRO)
	1	No output during type 5 block searches (SERUPRO)

Output counter

The user can output the collected auxiliary functions to the PLC on a channel-by-channel basis in the block search ASUB. For the purposes of serialized output via multiple channels, the three output counters are changed across all the channels each time an auxiliary function is output:

$AC_AUXFU_TICK[<n>,<m>] (output counter for the active auxiliary function)

<n>: Group index (0 to 63)

<m>: Output counter (0 ... 2)

- 0: Output sequence counter (all outputs within an IPO cycle)
- 1: Package counter within an output sequence in the IPO cycle
- 2: Auxiliary function counter within a package

Explanation:

An auxiliary function package comprises a maximum of ten auxiliary functions.

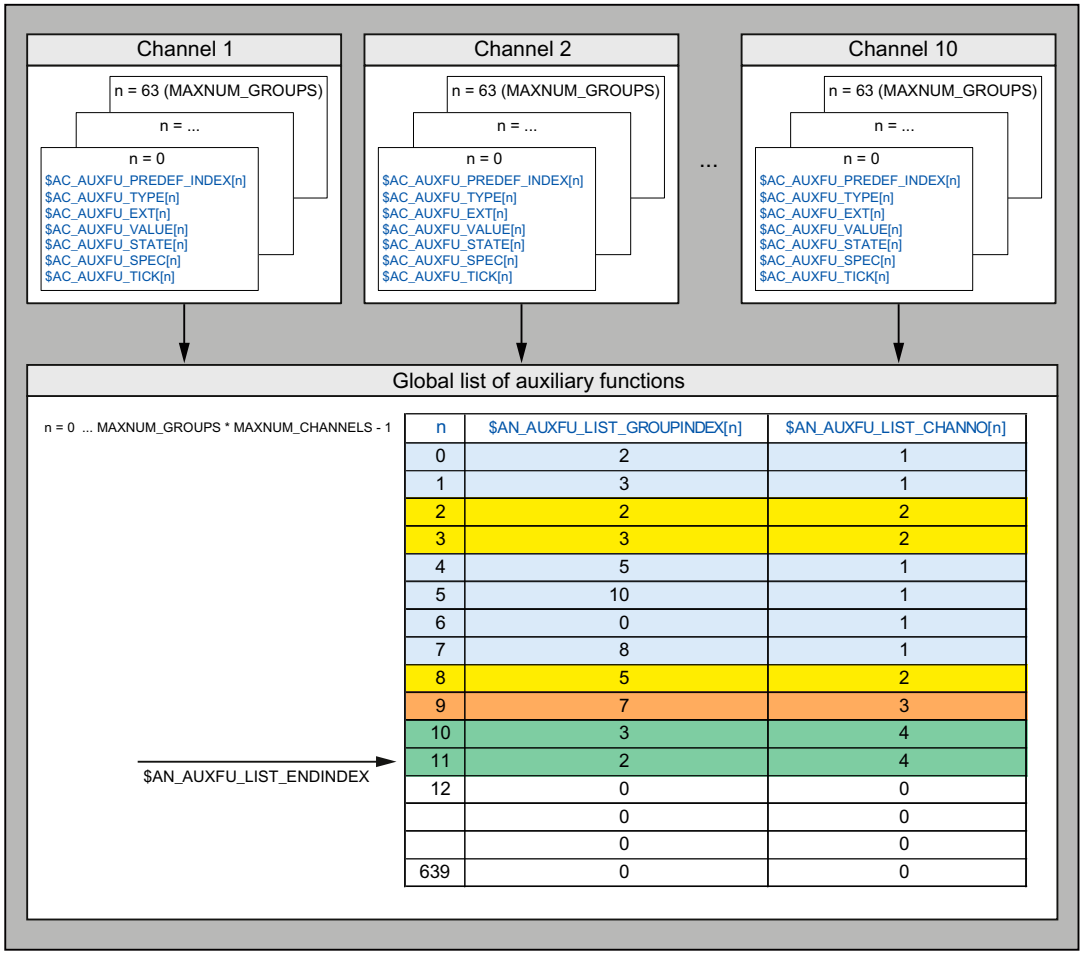
Two packages can be processed per IPO cycle in each channel during SERUPRO because synchronized actions are processed in this cycle.

An output sequence of up to a maximum of 20 packages (2 packages per channel * 10 channels) can be processed within an IPO cycle across all channels.

This encoding defines how many auxiliary function packages and, within these, how many auxiliary functions have been processed during the same IPO cycle: Auxiliary functions which have been collected in one IPO cycle all have the same sequence counter. Auxiliary functions which have been collected in one package (block or synchronized action) all have the same package counter. The total on the auxiliary function counter increases every time an auxiliary function is collected.

Global list of auxiliary functions

At the end of SERUPRO, the auxiliary functions collected on a group-specific basis in the individual channels are entered in a cross-channel (global) list with the channel number ($AN_AUXFU_LIST_CHANNO[<n>]) and group index ($AN_AUXFU_LIST_GROUPINDEX[<n>]) according to their counter state ($AC_AUXFU_TICK[<n>,<m>]).



n	$AN_AUXFU_LIST_GROUPINDEX[n]	$AN_AUXFU_LIST_CHANNO[n]
0	2	1
1	3	1
2	2	2
3	3	2
4	5	1
5	10	1
6	0	1
7	8	1
8	5	2
9	7	3
10	3	4
11	2	4
12	0	0
	0	0
	0	0
639	0	0

The two field variables $AN_AUXFU_LIST_CHANNO[<n>] and $AN_AUXFU_LIST_GROUPINDEX[<n>] from the global list are writable and readable. The index <n> has the following range of values: 0 ... MAXNUM_GROUPS * MAXNUM_CHANNELS - 1

The global list is structured on the basis of the sequence in which the search target was found. It is intended to be used as a system proposal for auxiliary functions to be output in the following ASUB at the end of SERUPRO. If an auxiliary function is not to be output, the corresponding group index is to be set to "0".

Behavior regarding spindle auxiliary functions

Following the start of the search, all the channels collect the auxiliary functions in the channel variables on a group-specific basis. In order to perform a far-reaching restoration of the spindle state in the SERUPRO target block using the collected auxiliary functions, the last active auxiliary function in any group of spindle auxiliary functions must characterize the state of the spindle in the target block. In the case of transitions in spindle states, obsolete auxiliary functions are deleted or, if necessary, implicit auxiliary functions are entered.

All the spindle auxiliary functions from the global auxiliary function list must correspond to the spindle states achieved in the target block to enable the auxiliary functions to be processed when the list is output and to ensure that no alarms or unintended spindle states are requested which could prevent the continuation of the part program.

This affects the groups of auxiliary functions for any spindle configured in the system, whereby the spindle number corresponds to an auxiliary function's address extension.

Group a:	M3, M4, M5, M19, M70
Group b:	M40, M41, M42, M43, M44, M45
Group c:	S

Deleting obsolete auxiliary functions

In the functions below, the auxiliary functions from group a are deleted for the spindle concerned:

- For the following spindle when a generic coupling, such as COUPON, TRAILON, EGON, etc. is switched on

Generating implicit auxiliary functions from group a

In the functions below, the auxiliary functions from group a are generated implicitly for the spindle concerned:

- For the following spindle when the synchronous spindle coupling is switched off
  - COUPOF generates M3, M4 and S or M5 in the main run depending on the coupling situation.
  - COUPOF(S<n>, S<m>, POS) and COUPOFS(S<n>, S<m>, POS, POS) generate M3, M4 and S.
  - COUPOFS generates M5 in the main run.
  - COUPOFS(S<n>, S<m>, POS) generates M19 in the main run.

    The implicit M19 ("SPOS[<address extension>] = IC(0)" in the ASUB) activates the positioning mode without a traversing motion.
- M70 is generated during a traversing motion as an axis or during the transition to axis mode by selecting a transformation during which the spindle enters as an axis.
- M5 is generated during SPCOF.

Note

Within the context of the "axis interchange" and "axis container rotation" functions, the auxiliary functions for programming the spindle must always be specified in a way which ensures compatibility with the actual (motor) state during interchange/rotation. A distinction is made here between the axis interchange and axis container mechanisms.

Example of axis container rotation:

An axis container has four spindles, each assigned to a separate channel (1 - 4). M3 S1000 is always programmed in channel 1, and an axis container rotation is then executed. The other channels do not perform any spindle programming. Following the third axis container rotation and the fourth spindle programming, M3, all four spindles rotate clockwise at a speed of 1,000 rpm. If the end of the SERUPRO now lies within this range, every ASUB for a channel is expected to contain an M3 S1000 for the local spindle.

During interchange however, the collected auxiliary functions may only be assigned to the channel where the spindle is currently located.

Cross-channel auxiliary function

An auxiliary function can also be collected on a cross-channel basis in the global auxiliary function list in the case of type 5 block searches 5 (SERUPRO). Only the last auxiliary function collected from this group (highest counter state) is entered in the global list.

Configuration takes place via bit 11 in the machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]

  (output behavior of predefined auxiliary functions)

  where <n> = system function index (0 ... 32)
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]

  (output behavior of user-defined auxiliary functions)

  where <n> = auxiliary function index (0 ... 254)
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]

  (output behavior of the auxiliary functions in a group)

  where <n> = group index (0 ... 63)

Bit	Value	Meaning
11	0	Channel-specific collection
	1	Cross-channel collection

The spindle auxiliary functions are filtered out beforehand at the end of the search depending on the spindle state. The channel data is updated accordingly. The global auxiliary function list can be processed in sequence in the ASUBs at the end of the SERUPRO, and the sorted auxiliary functions can be output with channel synchronization.

Querying the last auxiliary function collected

The index of the last auxiliary function collected in the global list can be queried using the system variable $AN_AUXFU_LIST_ENDINDEX.

7.13.7 ASUB at the end of the SERUPRO

Function

After completing the block search with the program test (SERUPRO), before starting the subsequent processing, the auxiliary functions collected during the search must be output. For this purpose, during the block search, the auxiliary functions are collected in a global list. The SERUPRO end ASUBs generate the corresponding part program blocks channel-specific from this list. This ensures that the collected auxiliary functions can be output both channel-specific as well as cross-channel in the correct sequence. A fully functional SERUPRO end ASUB is a component of the NCK software.

Users/machinery construction OEMs can change the SERUPRO end ASUB. The subsequently described functions support processing the global list of auxiliary functions and generating the part program blocks required for synchronized auxiliary function output.

Function AUXFUSYNC(...)

Function:

The function `AUXFUSYNC` generates a complete part program block as string from the global list of auxiliary functions at each call. The part program block either contains auxiliary functions or commands to synchronize auxiliary function outputs (`WAITM`, `G4`, etc.).

The function triggers a preprocessing stop.

Syntax:

```
PROC AUXFUSYNC(VAR INT <NUM>, VAR INT <GROUPINDEX>[10], VAR
STRING[400] <ASSEMBLED>)
```

Parameters:

<NUM>:	Contains information about the part program block, supplied in parameter <ASSEMBLED> or the auxiliary functions contained in it. Value range: -1, 0, 1 ... 10 **Value Meaning** ≥1 Number of auxiliary functions contained in the part program block 0 Part program block without auxiliary functions, e.g. WAITM, G4 -1 End identifier. The global list of auxiliary functions has been completely processed for the actual channel.
<GROUP INDEX>:	Contains the indices of the auxiliary function groups contained in the part program block. With index = number of the auxiliary function group - 1
<ASSEMBLED>:	Contains the complete part program block for the channel-specific SERUPRO end ASUB as string.

Further information:

If auxiliary functions were collected via a synchronized action, two NC blocks are generated. One NC block to output the auxiliary functions. An executable NC block via which the NC block is transported to the main run to output the auxiliary functions:

1. Output of the auxiliary functions via synchronized action, e.g.: `WHEN TRUE DO M100 M102`
2. Executable NC block, e.g.: `G4 F0.001`

Function AUXFUDEL(...)

Function:

The function AUXFUDEL deletes the specified auxiliary function from the global list of auxiliary functions channel-specific for the calling channel. Deletion is realized by setting the corresponding group index ...`GROUPINDEX[n]` to 0.

The function must be called before calling `AUXFUSYNC`.

The function triggers a preprocessing stop.

Syntax:

```
PROC AUXFUDEL(CHAR <TYPE>, INT <EXTENSION>, REAL <VALUE>, INT <GROUP>)
```

Parameters:

<TYPE>:	Type of auxiliary function to be deleted
<EXTENSION>:	Address extension of the auxiliary function to be deleted
<VALUE>:	Value of auxiliary function to be deleted
<GROUP>:	Number of the auxiliary function group

Function AUXFUDELG(...)

Function:

The function AUXFUDELG deletes all auxiliary functions of the specified auxiliary function group from the global list of auxiliary functions channel-specific for the calling channel. Deletion is realized by setting the corresponding group index ...`GROUPINDEX[n]` to 0.

The function must be called before calling `AUXFUSYNC`.

The function triggers a preprocessing stop.

Syntax:

```
PROC AUXFUDELG(INT <GROUP>)
```

Parameters:

```
<GROUP>:        Number of the auxiliary function group
```

Multi-channel block search

CAUTION
Multi-channel block search and AUXFUDEL / AUXFUDELG If, for a multi-channel block search in the SERUPRO end ASUBs, auxiliary functions with AUXFUDEL / AUXFUDELG are deleted from the global list of auxiliary functions, before calling the AUXFUSYNC function, the channels involved must be synchronized. The synchronization ensures that before calling the AUXFUSYNC, all delete requests are processed and a consistent list is available.

Examples

Two examples for configuring a user-specific SERUPRO end ASUB.

Example 1: Deleting auxiliary functions and generating the auxiliary function output with AUXFUSYNC(...)

```
Program code                                          Comment
N10 DEF STRING[400] ASSEMBLED=""
N20 DEF STRING[31] FILENAME="/_N_CST_DIR/_N_AUXFU_SPF"
N30 DEF INT GROUPINDEX[10]
N40 DEF INT NUM
N60 DEF INT ERROR
N90
N140 AUXFUDEL("M",2,3,5)                              ; M2=3 (5th auxiliary function group) delete
N150
N170 AUXFUDELG(6)                                     ; Delete the collected auxiliary function of the
                                                      ; 6. group.
N180
N190 IF ISFILE(FILENAME)
N210 DELETE(ERROR,FILENAME)                           ; Delete the FILENAME file
N220 IF (ERROR<>0)                                    ; Error evaluation
N230 SETAL(61000+ERROR)
N240 ENDIF
N250 ENDIF
; CAUTION!
; If, for a multi-channel block search, auxiliary functions with AUXFUDEL/AUXFUDELG
; are deleted from the global list of auxiliary functions, before the loop to
; generate the subprogram FILENAME with AUXFUSYNC, the channels must be synchronized.
 The synchronization ensures that all delete requests were processed
; in all channels and a consistent list is available.
; Example: WAITM(99,1,2,3)
N270 LOOP
N300 AUXFUSYNC(NUM,GROUPINDEX,ASSEMBLED)              ; General a part program block
N310
N320 IF (NUM==-1)                                     ; All auxiliary functions of the channel
                                                      ; have been executed.
N340 GOTOF LABEL1
N350 ENDIF
N380 WRITE(ERROR,FILENAME,ASSEMBLED)                  ; Write a part program block to file FILENAME.
N390 IF (ERROR<>0)                                    ; Error evaluation
N400 SETAL(61000+ERROR)
N410 ENDIF
N430 ENDLOOP
N440
N450 LABEL1:
```

Program code	Comment
N460	
N480 CALL FILENAME	; Process a generated subroutine.
N490	
N510 DELETE(ERROR,FILENAME)	; Delete the file again after execution.
N520 IF (ERROR<>0)	
N530 SETAL(61000+ERROR)	
N540 ENDIF	
N550	
N560 M17	

Example 2: Deleting auxiliary functions and generating the auxiliary function output without AUXFUSYNC(...)

Program code	Comment
N0610 DEF STRING[400] ASSEMBLED=""	
N0620 DEF STRING[31] FILENAME="/_N_CST_DIR/_N_AUXFU_SPF"	
N0630 DEF INT GROUPINDEX[10]	
N0640 DEF INT NUM	
N0650 DEF INT LAUF	
N0660 DEF INT ERROR	
N0670 DEF BOOL ISQUICK	
N0680 DEF BOOL ISSYNACT	
N0690 DEF BOOL ISIMPL	
...	
N0760 AUXFUDEL("M",2,3,5)	; M2=3 (5th auxiliary function group) delete
N0770	
N0790 AUXFUDELG(6)	; Delete the collected auxiliary function of the
	; 6. group.
N0800	
N0810 IF ISFILE(FILENAME)	
N0830 DELETE(ERROR,FILENAME)	; File already exists and must be
	; deleted.
N0840 IF (ERROR<>0)	
N0850 SETAL(61000+ERROR)	
N0860 ENDIF	
N0870 ENDIF	
N0880	
; CAUTION!	
; If, for a multi-channel block search, auxiliary functions with AUXFUDEL/AUXFUDELG	
; are deleted from the global list of auxiliary functions, before the loop to	
; generate the subprogram FILENAME with AUXFUSYNC, the channels must be synchronized.	
The synchronization ensures that all delete requests were processed	
; in all channels and a consistent list is available.	

```
; Example: WAITM(99,1,2,3)
N0890 LOOP
N0920 AUXFUSYNC(NUM,GROUPINDEX,ASSEMBLED)          ; Procedure to generate
                                                   ; auxiliary function blocks from the global
                                                   ; auxiliary function list.
N0930
N0940 IF (NUM==-1)                                 ; All auxiliary functions of the channel
                                                   ; are processed.
N0960 GOTOF LABEL1
N0970 ENDIF
N0980
N1000 IF (NUM>0)                                   ; If auxiliary functions are output,
                                                   ; the block is generated.
N1010
N1020 ASSEMBLED=""
N1030
N1050 FOR LAUF=0 TO NUM-1                          ; Collected auxiliary functions for a
                                                   ; block.
N1060
N1080 IF GROUPINDEX[LAUF]<>0                       ; Auxiliary functions deleted from the
                                                   ; global list have the group index 0.
N1090
N1100 ISQUICK=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H2'
N1110
N1120 ISSYNACT=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H1000'
N1130
N1140 ISIMPL=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H2000'
N1150
N1180 IF ISSYNACT                                  ; Assemble a block for the M auxiliary
                                                   ; function output
N1190 ASSEMBLED= ASSEMBLED << "WHEN TRUE DO "
N1200 ENDIF
N1210 ; Implicitly generated M19 is mapped to SPOS[SPI(<spindle no.>)] = IC(0).
N1230 IF (ISIMPL AND ($AC_AUXFU_VALUE[GROUPINDEX[LAUF]==19))
N1240 ASSEMBLED= ASSEMBLED << "SPOS[SPI(" <<
 $AC_AUXFU_EXT[GROUPINDEX[LAUF]] << ")=IC(0)"
N1260 ELSE
N1270 ASSEMBLED= ASSEMBLED << "M[" << $AC_AUXFU_EXT[GROUPINDEX[LAUF]] << "]="
N1280
N1290 IF ISQUICK
N1300 ASSEMBLED= ASSEMBLED << "QU("
N1310 ENDIF
N1320
N1330 ASSEMBLED= ASSEMBLED << $AC_AUXFU_VALUE[GROUPINDEX[LAUF]]
```

Program code	Comment
N1340	
N1350 IF ISQUICK	
N1360 ASSEMBLED= ASSEMBLED << ")"	
N1370 ENDIF	
N1380 ENDIF	
N1400 ENDIF	
N1420 ENDFOR	
N1430	
N1450 WRITE(ERROR,FILENAME,ASSEMBLED)	; Write an auxiliary function block to a file.
N1460	
N1470 IF ISSYNACT	
N1480 ASSEMBLED="G4 F0.001"	
N1490 WRITE(ERROR,FILENAME,ASSEMBLED)	
N1500 ENDIF	
N1510	
N1520 ELSE	
N1540 WRITE(ERROR,FILENAME,ASSEMBLED)	; Write an auxiliary function block to a file.
N1550 ENDIF	
N1560	
N1570 ENDLOOP	
N1580	
N1590 LABEL1:	
N1600	
N1620 CALL FILENAME	; Process a generated subroutine.
N1630	
N1650 DELETE(ERROR,FILENAME)	; Delete the file again after execution.
N1660 IF (ERROR<>0)	
N1670 SETAL(61000+ERROR)	
N1680 ENDIF	
N1690	
N1700 M17	

7.14 Implicitly output auxiliary functions

Function

Implicitly output auxiliary functions are auxiliary functions which have not been programmed explicitly and which are also output by other system functions (e.g. transformation selection, tool selection, etc.). These implicit auxiliary functions do not lead to any system function; instead, the M codes are collected according to output behavior parameters assigned to them and/or are output to the PLC.

Parameterization

The M codes for auxiliary functions to be output implicitly are defined with the machine data:

- MD22530 $MC_TOCARR_CHANGE_M_CODE (M code at toolholder change)

  This machine data value indicates the number of the M code which is output when a toolholder is activated at the NC/PLC interface.

  If the value is positive, the unchanged M code is always output.

  If the value is negative, the number of the toolholder is added to the machine data value, and this number is output.

- MD22532 $MC_GEOAX_CHANGE_M_CODE (M code when switching the geometry axes)

  Number of the M code which is output when the geometry axes on the NC/PLC interface are switched.

- MD22534 $MC_TRAFO_CHANGE_M_CODE (M code in the case of transformation changes)

  Number of the M code which is output during a transformation switch of the geometry axes at the NC/PLC interface.

Note

No M code is output if the number of the M code being output or the MD22530/MD22532/MD22534 value is between 0 and 6, or is either 17 or 30. Whether or not an M code which is generated in this manner leads to conflicts with other functions is not monitored.

Output behavior

In the case of implicitly output auxiliary functions, bit 13 is set in machine data MD22080 or MD22035 (output behavior of predefined or user-defined auxiliary functions).

This bit can be queried via the system variable $AC_AUXFU_SPEC[<n>].

Implicitly output auxiliary function M19

To achieve uniformity in terms of how M19 and SPOS or SPOSA behave at the NC/PLC interface, auxiliary function M19 can be output to the NC/PLC interface in the event of SPOS and SPOSA (see Section "General functionality (Page 1390)").

The implicitly output auxiliary function M19 is collected during the block search.

7.15 Information options

Information about auxiliary functions (e.g. about the output status) is possible via:

- The group-specific modal M auxiliary function display on the user interface
- Querying system variables in part programs and synchronized actions

7.15.1 Group-specific modal M auxiliary function display

Function

The output status and acknowledgement status of M auxiliary functions can be displayed on the user interface on a group-specific basis.

Requirements

To implement function-oriented acknowledgement and display of M auxiliary functions, the auxiliary functions must be managed in the PLC and, thus, in the user program itself. Therefore, it is up to the PLC programmer to program the acknowledgement of these auxiliary functions. He has to know which auxiliary functions in which group have to be acknowledged.

Standard

M auxiliary functions that are not managed by means of the PLC are identified by the NC as "transferred" and output to the PLC. There is no functional acknowledgement for these auxiliary functions. All M-auxiliary functions collected after a block search are also displayed so that the operator knows which auxiliary functions will be output after a start following a block search.

PLC activities

In the case of auxiliary function groups that are managed by the PLC itself, the PLC user program must acknowledge all auxiliary functions of this groups when **Apply** and **Function End** are activated. The PLC programmer must know all the auxiliary functions of these groups.

Miscellaneous

Only the **group-specific** M auxiliary functions are displayed. The block-by-block display is also retained. Up to 15 groups can be displayed, whereby **only the last M function of a group** that was either collected or output to the PLC is displayed for each group. The M functions are presented in various display modes depending on their status:

Status	Display mode
Auxiliary function is collected	Inverted with yellow font
Auxiliary function is output from NCK to PLC	Inverted
Auxiliary function has been transferred from NCK to PLC and transport acknowledgement has taken place	Black font on gray background
Auxiliary function is managed by the PLC and has been directly applied by the PLC.	Black font on gray background
Auxiliary function is managed by the PLC, and the function acknowledgement has taken place.	Black font on gray background

Display update

The display is organized in such a way that the collected auxiliary functions are always displayed first, before those that were managed by the PLC and before those that were managed by the NC. A collected auxiliary function is marked as collected until it has been output from the NCK to the PLC. PLC-managed auxiliary functions are retained until they are displaced by another auxiliary function. In the case of a reset, only the collected auxiliary functions and the NC-managed auxiliary functions are deleted.

7.15.2 Querying system variables

Function

Auxiliary functions can be queried on a group-specific basis via system variables in the part program and via synchronized actions:

$AC_AUXFU_... [<n>] = <value>

system variables	Meaning		
$AC_AUXFU_PREDEF_INDEX[<n>]	<value>:	Index of the last auxiliary function collected for an auxiliary function group (search) or the last predefined auxiliary function to be output	
		Type:	INT
		If no auxiliary function has been output yet for the specified group or if the auxiliary function is a user-defined auxiliary function, the variable supplies the value "-1".	
	<n>:	Group index (0 to 63)	
	Note: A predefined auxiliary function can be uniquely identified via this variable.		
$AC_AUXFU_TYPE[<n>]	<value>:	Type of the last auxiliary function collected for an auxiliary function group (search) or the last auxiliary function to be output	
		Type:	CHAR
	<n>:	Group index (0 to 63)	
$AC_AUXFU_EXT[<n>] or M function specific: $AC_AUXFU_M_EXT[<n>]	<value>:	Address extension of the last auxiliary function collected for an auxiliary function group (search) or the last auxiliary function to be output	
		Type:	INT
	<n>:	Group index (0 to 63)	
$AC_AUXFU_VALUE[<n>] or M function specific: $AC_AUXFU_M_VALUE[<n>]	<value>:	Value of the last auxiliary function collected for an auxiliary function group (search) or the last auxiliary function to be output	
		Type:	REAL
	<n>:	Group index (0 to 63)	
$AC_AUXFU_SPEC[<n>]	Value:	Bit-encoded output behavior according to MD22080/MD22035 (or QU programming) of the last auxiliary function collected for an auxiliary function group (search) or the last auxiliary function to be output	
		Type:	INT
	<n>:	Group index (0 to 63)	

system variables	Meaning		
	Note: This variable can be used to determine whether the auxiliary function should be output with a fast acknowledgement.		
$AC_AUXFU_STATE[<n>] or M function specific: $AC_AUXFU_M_STATE[<n>]	<value>:	Output state of the last auxiliary function collected for an auxiliary function group (search) or the last auxiliary function to be output	
		Type:	INT
		Value range:	0 ... 5
		0:	No auxiliary function
		1:	M-auxiliary function was collected via a search
		2:	M-auxiliary function has been output to the PLC
		3:	M-auxiliary function has been output to the PLC and the transport acknowledgement has taken place
		4:	M-auxiliary function is managed by the PLC and has been applied by the PLC
		5:	M-auxiliary function is managed by the PLC, and the function acknowledgement has taken place
	<n>:	Group index (0 to 63)	

Example

All M-auxiliary functions of the 1st group will be stored in the order they are output

```
id=1 every $AC_AUXFU_M_STATE[0]==2 do
$AC_FIFO[0,0]=$AC_AUXFU_M_VALUE[0]
```

References

For more information on the system variables, refer to:

List Manual, system variables

7.16 Supplementary conditions

7.16.1 General constraints

Spindle replacement

Because the auxiliary functions are parameterized channel-specifically, if function: "spindle replacement" is used, the spindle-specific auxiliary function must be parameterized immediately in all channels that use the spindles.

Tool management

If tool management is active, the following constraints apply:

- T and M<k> functions are not output to the PLC.
  Note
  k is the parameterized value of the auxiliary function for the tool change (default: 6):
  MD22560 $MC_TOOL_CHANGE_M_CODE (auxiliary function for tool change)
- If no address extension is programmed, the auxiliary function refers to the master spindle or the master tool holder of the channel.

  Definition of the master spindle:

  – MD20090 $MC_SPIND_DEF_MASTER_SPIND

  – Part program instruction: SETMS

  Definition of the master tool holder

  – MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER

  – Part program instruction: SETMTH

Maximum number of auxiliary functions per part program block

A maximum of 10 auxiliary functions may be programmed in one part program block.

DL (additive tool offset)

The following restrictions apply to the DL function:

- Only one DL function can be programmed per part program block.
- If DL functions are used in synchronous actions, parameter: "Value" is not output to the PLC.

7.16.2 Output behavior

Thread cutting

During active thread cutting G33, G34 and G35, the following output behavior is always active for the spindle-specific auxiliary functions:

- M3 (spindle right)
- M4 (spindle left)

- Output duration of one OB40 cycle (quick acknowledgement)
- Output during motion

The spindle-specific auxiliary function M5 (spindle stop) is always output at the end of the block. The part program block that contains M5 is always ended with exact stop, i.e. even during active continuous-path mode.

Synchronized actions

With output auxiliary functions from synchronized actions, the parameterized output behavior is ignored except for the following parameters:

- Bit0: Output duration of one OB1 cycle (normal acknowledgement)
- Bit1: Output duration of one OB40 cycle (quick acknowledgement)

Auxiliary functions: M17 or M2/M30 (end of subprogram)

In its own part program block

If one of the auxiliary functions M17, M2 or M30 is programmed as the only auxiliary function in a part program block and an axis is still in motion, the auxiliary function is not output to the PLC until after the axis has stopped.

Overriding the parameterized output behavior

The parameterized output behavior of the auxiliary functions M17 or M2/M30 is overridden by the output behavior that is determined in the following machine data:

MD20800 $MC_SPF_END_TO_VDI, bit 0 (subprogram end / stop to PLC)

Bit	Value	Meaning
0	0	The auxiliary functions M17 or M2/M30 (subprogram end) are not output to the PLC. Continuous-path mode is not interrupted at the end of the subprogram
	1	The auxiliary functions M17 or M2/M30 (subprogram end) are output to the PLC.

Auxiliary function: M1 (conditional stop)

Overriding the parameterized output behavior

The parameterized output behavior of the auxiliary function M1 is overridden by the output behavior defined in the following machine data:

MD20800 $MC_SPF_END_TO_VDI, bit 1 (subprogram end / stop to PLC)

Bit	Value	Meaning
1	0	The auxiliary function M01 (conditional stop) is always output to the PLC. A quick acknowledgement is ineffective, because M01 is permanently assigned to the first auxiliary function group and is therefore always output at the end of the block.
	1	The auxiliary function M01 (conditional stop) is only output to the PLC, if the function: "Programmed stop" is activated via the HMI user interface. In the case of a quick acknowledgement, the M1 is output to the PLC during the motion. While the function is not active, this does not interrupt continuous-path mode.

Part program blocks without traversing motion

In a part program block without a traversing motion, all auxiliary functions are output in a block immediately, irrespective of their parameterized output behavior.

Spindle-specific auxiliary function output only as information for the PLC user program

In certain controller situations, e.g. at the end of a block search, the collected spindle-specific auxiliary functions (e.g. M3, M4, M5, M19, M40...M45, M70) is output to the NC/PLC interface only for information purposes for the PLC user program. The controller generates a part program block (action block) in which the collected auxiliary functions are entered with a **negative** address extension. The corresponding system functions are then **not** executed.

Example: M(-2) = 41 request gear stage change for the 2nd spindle

7.17 Examples

7.17.1 Extension of predefined auxiliary functions

Task

Parameter assignment of auxiliary functions M3, M4, and M5 for the second spindle of the channel

Parameter assignment: M3

Requirements:

- Machine data index: 0 (first user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M3 (spindle right)
- Address extension: 2 as appropriate for the 2nd spindle of the channel
- Output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Output prior to motion

Parameter assignment:

```
MD22000 $MC_AUXFU_ASSIGN_GROUP[ 0 ]          = 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 0 ]          = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 0 ]     = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 0 ]         = 3
MD22035 $MC_AUXFU_ASSIGN_SPEC [ 0 ]          = 'H21'
```

Parameter assignment: M4

Requirements:

- Machine data index: 1 (second user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M4 (spindle left)
- Address extension: 2 as appropriate for the 2nd spindle of the channel
- Output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Spindle response following acknowledgment
  - Output during motion

Parameter assignment:

```
MD22000 $MC_AUXFU_ASSIGN_GROUP [ 1 ]        = 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 1 ]         = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 1 ]    = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 1 ]        = 4
MD22035 $MC_AUXFU_ASSIGN_SPEC [ 1 ]         = 'H51'
```

Parameter assignment: M5

Requirements:

- Machine data index: 2 (third user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M5 (spindle stop)
- Address extension: 2 as appropriate for the 2nd spindle of the channel
- Output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Spindle response following acknowledgment
  - Output at block end

Parameter assignment:

```
MD22000 $MC_AUXFU_ASSIGN_GROUP [ 2 ]        = 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 2 ]         = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 2 ]    = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 2 ]        = 5
MD22035 $MC_AUXFU_ASSIGN_SPEC [ 2 ]         = 'H91'
```

7.17.2 Defining auxiliary functions

Task

Parameter assignment of the auxiliary function-specific machine data for a machine with the following configuration:

Spindles

- Spindle 1: Master spindle
- Spindle 2: Second spindle

Gear stages

- Spindle 1: 5 gear stages
- Spindle 2: No gear stages

Switching functions for cooling water on/off

- Spindle 1
  - "ON" = M50
  - "OFF" = M51
- Spindle 2
  - "ON" = M52
  - "OFF" = M53

Requirements

Spindle 1 (master spindle)

Note

Default assignments

- The auxiliary functions M3, M4, M5, M70 and M1=3, M1=4, M1=5, M1=70 of spindle 1 (master spindle) are assigned as standard to the second auxiliary function group.
- All S and S1 values of spindle 1 (master spindle) are assigned to the third auxiliary function group by default.

- The gear stage last programmed is to be output after block search. The following auxiliary functions are assigned to the ninth auxiliary function group for this reason:
  - M40, M41, M42, M43, M44, M45
  - M1=40, M1=41, M1=42, M1=43, M1=44, M1=45
- The auxiliary functions M3, M4, M5, M70 and M1=3, M1=4, M1=5, M1=70 (second auxiliary function group) and S and S1 values (third auxiliary function group) should possess the following output behavior:
  - Output duration one OB40 cycle (quick acknowledgment)
  - Output prior to motion
- The auxiliary functions for gear changeover M40 to M45 and M1=40 to M1=45 (ninth auxiliary function group) should have the following output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Output prior to motion

Spindle 2

- Only one M function for directional reversal may be programmed in one block. The direction of rotation last programmed is to be output after block search. The following auxiliary functions are assigned to the tenth auxiliary function group for this reason:
  - M2=3, M2=4, M2=5, M2=70
- All S2 values are assigned to auxiliary function group 11.
- The auxiliary functions M2=3, M2=4, M2=5, M2=70 (tenth auxiliary function group) and S2 values (auxiliary function group 11) should have the following output behavior:
  - Output duration one OB40 cycle (quick acknowledgment)
  - Output prior to motion

Cooling water

- It is not permissible to switch the cooling water on and off in one part program block. After a block search, the cooling water will be switched on or off. For this purpose, the following auxiliary functions are assigned, for example, to auxiliary function group 12 or 13:
  - Auxiliary function group 12: M50, M51
  - Auxiliary function group 13: M52, M53
- The auxiliary functions M50, M51 (auxiliary function group 12) and M52, M53 (auxiliary function group 13) should have the following output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Output prior to motion

Parameterization of the machine data

The machine data are parameterized by appropriate programming within a parts program.

Program code	Comment
$MN_AUXFU_MAXNUM_GROUP_ASSIGN=21	; Number of user-defined auxiliary functions per channel
$MN_AUXFU_GROUP_SPEC[1]='H22'	; Output behavior of auxiliary function group 2
$MN_AUXFU_GROUP_SPEC[2]='H22'	; Output behavior of auxiliary function group 3
$MN_AUXFU_GROUP_SPEC[8]='H21'	; Output behavior of auxiliary function group 9
$MC_AUXFU_ASSIGN_TYPE[0]="M"	; Description of auxiliary function 1: M40
$MC_AUXFU_ASSIGN_EXTENSION[0]=0	
$MC_AUXFU_ASSIGN_VALUE[0]=40	
$MC_AUXFU_ASSIGN_GROUP[0]=9	
	; ... (and analogously for aux. functions 2 to 5)
$MC_AUXFU_ASSIGN_TYPE[5]="M"	; Description of auxiliary function 6: M45
$MC_AUXFU_ASSIGN_EXTENSION[5]=0	
$MC_AUXFU_ASSIGN_VALUE[5]=45	
$MC_AUXFU_ASSIGN_GROUP[5]=9	
$MC_AUXFU_ASSIGN_TYPE[6]="M"	; Description of auxiliary function 7: M1=40
$MC_AUXFU_ASSIGN_EXTENSION[6]=1	
$MC_AUXFU_ASSIGN_VALUE[6]=40	
$MC_AUXFU_ASSIGN_GROUP[6]=9	
	; . . . (and analogously for aux. functions 8 to 11)
$MC_AUXFU_ASSIGN_TYPE[11]="M"	; Description of auxiliary function 12: M1=45
$MC_AUXFU_ASSIGN_EXTENSION[11]=1	
$MC_AUXFU_ASSIGN_VALUE[11]=45	
$MC_AUXFU_ASSIGN_GROUP[11]=9	
$MN_AUXFU_GROUP_SPEC[9] = 'H22'	; Output behavior of auxiliary function group 10
$MC_AUXFU_ASSIGN_TYPE[12]="M"	; Description of auxiliary function 13: M2=3
$MC_AUXFU_ASSIGN_EXTENSION[12]=2	
$MC_AUXFU_ASSIGN_VALUE[12]=3	
$MC_AUXFU_ASSIGN_GROUP[12]=10	
$MC_AUXFU_ASSIGN_TYPE[13]="M"	; Description of auxiliary function 14: M2=4
$MC_AUXFU_ASSIGN_EXTENSION[13]=2	
$MC_AUXFU_ASSIGN_VALUE[13]=4	
$MC_AUXFU_ASSIGN_GROUP[13]=10	

Program code	Comment
$MC_AUXFU_ASSIGN_TYPE[14]="M"	; Description of auxiliary function 15: M2=5
$MC_AUXFU_ASSIGN_EXTENSION[14]=2	
$MC_AUXFU_ASSIGN_VALUE[14]=5	
$MC_AUXFU_ASSIGN_GROUP[14]=10	
$MC_AUXFU_ASSIGN_TYPE[15]="M"	; Description of auxiliary function 16: M2=70
$MC_AUXFU_ASSIGN_EXTENSION[15]=2	
$MC_AUXFU_ASSIGN_VALUE[15]=70	
$MC_AUXFU_ASSIGN_GROUP[15]=10	
$MN_AUXFU_GROUP_SPEC[10] = 'H22'	; Specification of auxiliary function group 11
$MC_AUXFU_ASSIGN_TYPE[16] = "S"	; Description of auxiliary function 17: S2=<all values>
$MC_AUXFU_ASSIGN_EXTENSION[16]=2	
$MC_AUXFU_ASSIGN_VALUE[16]=-1	
$MC_AUXFU_ASSIGN_GROUP[16]=11	
$MN_AUXFU_GROUP_SPEC[11]='H21'	; Specification of auxiliary function group 12
$MC_AUXFU_ASSIGN_TYPE[17]="M"	; Description of auxiliary function 18: M50
$MC_AUXFU_ASSIGN_EXTENSION[17]=0	
$MC_AUXFU_ASSIGN_VALUE[17]=50	
$MC_AUXFU_ASSIGN_GROUP[17]=12	
$MC_AUXFU_ASSIGN_TYPE[18]="M"	; Description of auxiliary function 19: M51
$MC_AUXFU_ASSIGN_EXTENSION[18]=0	
$MC_AUXFU_ASSIGN_VALUE[18]=51	
$MC_AUXFU_ASSIGN_GROUP[18]=12	
$MN_AUXFU_GROUP_SPEC[12]='H21'	; Specification of auxiliary function group 13
$MC_AUXFU_ASSIGN_TYPE[19]="M"	; Description of auxiliary function 20: M52
$MC_AUXFU_ASSIGN_EXTENSION[19]=0	
$MC_AUXFU_ASSIGN_VALUE[19]=52	
$MC_AUXFU_ASSIGN_GROUP[19]=13	
$MC_AUXFU_ASSIGN_TYPE[20]="M"	; Description of auxiliary function 21: M53
$MC_AUXFU_ASSIGN_EXTENSION[20]=0	
$MC_AUXFU_ASSIGN_VALUE[20]=53	
$MC_AUXFU_ASSIGN_GROUP[20]=13	

7.18 Data lists

7.18.1 Machine data

7.18.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10713	M_NO_FCT_STOPRE	M function with preprocessing stop
10714	M_NO_FCT_EOP	M function for spindle active after NC RESET
10715	M_NO_FCT_CYCLE	M function to be replaced by subroutine
11100	AUXFU_MAXNUM_GROUP_ASSIGN	Maximum number of user-defined auxiliary functions per channel
11110	AUXFU_GROUP_SPEC	Group-specific output behavior
11450	SEARCH_RUN_MODE	Behavior after a block search

7.18.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control initial setting after part program start.
20112	START_MODE_MASK	Definition of control initial setting after powerup and on RESET or at end of part program
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20800	SPF_END_TO_VDI	Subroutine end / Stop at PLC
22000	AUXFU_ASSIGN_GROUP	Group assignment of user-defined auxiliary functions
22010	AUXFU_ASSIGN_TYPE	Type of user-defined auxiliary functions
22020	AUXFU_ASSIGN_EXTENSION	Address extension for user-defined auxiliary functions
22030	AUXFU_ASSIGN_VALUE	Value of user-defined auxiliary functions
22035	AUXFU_ASSIGN_SPEC	Output behavior of user-defined auxiliary functions
22040	AUXFU_PREDEF_GROUP	Group assignment of predefined auxiliary functions
22050	AUXFU_PREDEF_TYPE	Type of predefined auxiliary functions
22060	AUXFU_PREDEF_EXTENSION	Address extension for predefined auxiliary functions
22070	AUXFU_PREDEF_VALUE	Value of predefined auxiliary functions
22080	AUXFU_PREDEF_SPEC	Output behavior of predefined auxiliary functions

Number	Identifier: $MC_	Description
22100	AUXFU_QUICK_BLOCKCHANGE	Block change delay with quick auxiliary functions
22110	AUXFU_H_TYPE_INT	Type of H auxiliary functions
22200	AUXFU_M_SYNC_TYPE	M functions output time
22210	AUXFU_S_SYNC_TYPE	S functions output time
22220	AUXFU_T_SYNC_TYPE	T functions output time
22230	AUXFU_H_SYNC_TYPE	H functions output time
22240	AUXFU_F_SYNC_TYPE	F functions output time
22250	AUXFU_D_SYNC_TYPE	D functions output time
22252	AUXFU_DL_SYNC_TYPE	DL functions output time
22254	AUXFU_ASSOC_M0_VALUE	Additional M function for program stop
22256	AUXFU_ASSOC_M1_VALUE	Additional M function for conditional stop
22530	TOCARR_CHANGE_M_CODE	M code for change of tool holder
22532	GEOAX_CHANGE_M_CODE	M code for replacement of geometry axes
22534	TRAFO_CHANGE_M_CODE	M code for change of tool holder
22560	TOOL_CHANGE_M_CODE	Auxiliary function for tool change

7.18.2 Signals

7.18.2.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate associated M01	DB21, ...DBX30.5	DB3200.DBX14.5

7.18.2.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
M function 1 - 5 change	DB21, ...DBX58.0-4	DB2500.DBX4.0-4
M function 1 - 5 not decoded	DB21, ...DBX59.0-4	-
S function 1 - 3 change	DB21, ...DBX60.0-2	DB2500.DBX6.0
S function 1 - 3 quick	DB21, ...DBX60.4-6	-
T function 1 - 3 change	DB21, ...DBX61.0-2	DB2500.DBX8.0
T function 1 - 3 quick	DB21, ...DBX61.4-6	-
D function 1 - 3 change	DB21, ...DBX62.0-2	DB2500.DBX10.0
D function 1 - 3 quick	DB21, ...DBX62.4-6	-
DL function change	DB21, ...DBX63.0	-
DL function quick	DB21, ...DBX63.4	-
H function 1 - 3 change	DB21, ...DBX64.0-2	DB2500.DBX12.0-2

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
H function 1 - 3 quick	DB21, ...DBX64.4-6	-
F function 1 - 6 change	DB21, ...DBX65.0-5	-
M function 1 - 5 quick	DB21, ...DBX66.0-4	-
F function 1 - 6 quick	DB21, ...DBX67.0-5	-
Extended address M function 1 (16 bit int)	DB21, ...DBB68-69	DB2500.DBB3004
M function 1 (DInt)	DB21, ...DBB70-73	DB2500.DBD3000
Extended address M function 2 (16 bit int)	DB21, ...DBB74-75	DB2500.DBB3012
M function 2 (DInt)	DB21, ...DBB76-79	DB2500.DBD3008
Extended address M function 3 (16 bit int)	DB21, ...DBB80-81	DB2500.DBB3020
M function 3 (DInt)	DB21, ...DBB82-85	DB2500.DBD3016
Extended address M function 4 (16 bit int)	DB21, ...DBB86-87	DB2500.DBB3028
M function 4 (DInt)	DB21, ...DBB88-91	DB2500.DBD3024
Extended address M function 5 (16 bit int)	DB21, ...DBB92-93	DB2500.DBB3036
M function 5 (DInt)	DB21, ...DBB94-97	DB2500.DBD3032
Extended address S function 1 (16 bit int)	DB21, ...DBB98-99	DB2500.DBB4004
S function 1 (real)	DB21, ...DBB100-103	DB2500.DBD4000
Extended address S function 2 (16 bit int)	DB21, ...DBB104-105	DB2500.DBB4012
S function 2 (real)	DB21, ...DBB106-109	DB2500.DBD4008
Extended address S function 3 (16 bit int)	DB21, ...DBB110-111	DB2500.DBB4020
S function 3 (real)	DB21, ...DBB112-115	DB2500.DBD4016
Extended address T function 1 (16 bit int)	DB21, ...DBB116-117	DB2500.DBB2004
T function 1 (integer)	DB21, ...DBB118-119	DB2500.DBD2000
Extended address T function 2 (16 bit int)	DB21, ...DBB120-121	-
T function 2 (integer)	DB21, ...DBB122-123	-
Extended address T function 3 (16 bit int)	DB21, ...DBB124-125	-
T function 3 (integer)	DB21, ...DBB126-127	-
Extended address D function 1 (8 bit int)	DB21, ...DBB128	DB2500.DBB5004
D function 1 (8 bit int)	DB21, ...DBB129	DB2500.DBD5000
Extended address D function 2 (8 bit int)	DB21, ...DBB130	-
D function 2 (8 bit int)	DB21, ...DBB131	-
Extended address D function 3 (8 bit int)	DB21, ...DBB132	-
D function 3 (8 bit int)	DB21, ...DBB133	-
Extended address DL function (8 bit int)	DB21, ...DBB134	-
DL function (real)	DB21, ...DBB136	-
Extended address H function 1 (16 bit int)	DB21, ...DBB140-141	DB2500.DBB6004
H function 1 (real or DInt)	DB21, ...DBB142-145	DB2500.DBD6000
Extended address H function 2 (16 bit int)	DB21, ...DBB146-147	DB2500.DBB6012
H function 2 (REAL or DInt)	DB21, ...DBB148-151	DB2500.DBD6008
Extended address H function 3 (16 bit int)	DB21, ...DBB152-153	DB2500.DBB6020
H function 3 (real or DInt)	DB21, ...DBB154-157	DB2500.DBD6016

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Extended address F function 1 (16 bit int)	DB21, ...DBB158-159	-
F function 1 (real)	DB21, ...DBB160-163	-
Extended address F function 2 (16 bit int)	DB21, ...DBB164-165	-
F function 2 (real)	DB21, ...DBB166-169	-
Extended address F function 3 (16 bit int)	DB21, ...DBB170-171	-
F function 3 (real)	DB21, ...DBB172-175	-
Extended address F function 4 (16 bit int)	DB21, ...DBB176-177	-
F function 4 (real)	DB21, ...DBB178-181	-
Extended address F function 5 (16 bit int)	DB21, ...DBB182-183	-
F function 5 (real)	DB21, ...DBB184-187	-
Extended address F function 6 (16 bit int)	DB21, ...DBB188-189	-
F function 6 (real)	DB21, ...DBB190-193	-
Dynamic M function: M00 - M07	DB21, ...DBB194	DB2500.DBB1000
Dynamic M function: M08 - M15	DB21, ...DBB195	DB2500.DBB1001
Dynamic M function: M16 - M23	DB21, ...DBB196	DB2500.DBB1002
Dynamic M function: M24 - M31	DB21, ...DBB197	DB2500.DBB1003
Dynamic M function: M32 - M39	DB21, ...DBB198	DB2500.DBB1004
Dynamic M function: M40 - M47	DB21, ...DBB199	DB2500.DBB1005
Dynamic M function: M48 - M55	DB21, ...DBB200	DB2500.DBB1006
Dynamic M function: M56 - M63	DB21, ...DBB201	DB2500.DBB1007
Dynamic M function: M64 - M71	DB21, ...DBB202	DB2500.DBB1008
Dynamic M function: M72 - M79	DB21, ...DBB203	DB2500.DBB1009
Dynamic M function: M80 - M87	DB21, ...DBB204	DB2500.DBB1010
Dynamic M function: M88 - M95	DB21, ...DBB205	DB2500.DBB1011
Dynamic M function: M96 - M99	DB21, ...DBX206.0-3	DB2500.DBB1012.0-3
Associated M00/M01 active	DB21, ...DBX318.5	DB3300.DBX4002.5

7.18.2.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
F function for positioning axis (real)	DB31, ...DBB78-81	-

7.18.2.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
M function for spindle (Int)	DB21, ...DBB86-87	DB370x.DBD0000
S function for spindle (real)	DB21, ...DBB88-91	DB370x.DBD0004

K1: Mode group, channel, program operation, reset response

8

8.1 Product brief

Channel

An NC channel represents the smallest unit for manual traversing of axes and automatic processing of part programs. At any one time, a channel will always be in a particular mode, e.g. AUTOMATIC, MDI, or JOG. A channel can be regarded as an independent NC.

Mode group

A channel always belongs to a mode group. A mode group can also consist of several channels.

A mode group can be identified by the fact that all channels of the mode group are always in the same mode at a particular time, e.g. AUTOMATIC, MDI, or JOG. This is ensured through the NC internal mode logic.

A mode group can be regarded as an independent multi-channel NC.

Channel gaps

When channels are configured, placeholder channels can be provided in order to create as uniform a configuration as possible over machines in a series. Only the channels that are actually used are then activated.

Program test

The following options are available for testing or moving in position a new part program.

- Program execution without setpoint outputs
- Program execution in singleblock mode
- Program execution with dry run feedrate
- Skip part program blocks
- Block search with or without calculation.

Block search

The block search function enables the following program simulations for locating specific program points:

- Type 1 without calculation at contour
- Type 2 with calculation at contour
- Type 4 with calculation at block end point
- Type 5 automatic start of the selected program point with calculation of all required data from history
- Automatic start of an ASUB after a block search
- Cascaded block search
- Cross-channel block search in "Program test" mode

Program operation

The execution of part programs or part program blocks in AUTOMATIC or MDI modes is referred to as program operation. During execution, the program sequence can be controlled by PLC interface signals and commands.

For each channel, basic settings or channel-specific machine data can be specified. These initial settings affect, for example, G groups and auxiliary function output.

A part program can be selected only if the relevant channel is in the Reset state.

Furthermore, all further program runs are handled by PLC interface signals and the corresponding commands.

- Start of part program or part program block
- Part program calculation and program control
- RESET command, program status, and channel status
- Responses to operator and program actions
- Eventdriven program calls

Asynchronous subroutines (ASUBs), interrupt routines

Interrupt inputs allow the NC to interrupt the the current part program execution so that it can react to more urgent events in interrupt routines or ASUBs.

Single block

With the single-block function, the user can execute a part program block-by-block.

There are 3 types of setting for the single-block function:

- SLB1: = IPO single block
- SLB2: = Decode single block
- SLB3: = Stop in cycle

Basic block display

A second basic block display can be used with the existing block display to display all blocks that produce an action on the machine.

The actually approached end positions are shown as an absolute position. The position values refer either to the workpiece coordinate system (WCS) or the settable zero system (SZS).

Program execution from external source

When complex workpieces are machined, the NC may not have enough memory for the programs. The "Execution from external source" function enables subroutines to be called (`EXTCALL`) and executed from an external memory (e.g. from the HMI Advanced hard disk).

Behavior after POWER ON, Reset, ...

The control-system response after:

- Power up (POWER ON)
- Reset/part program end
- Part program start

can be modified for functions, such as G codes, tool length compensation, transformation, coupled axis groupings, tangential follow-up, programmable synchronous spindle for certain system settings through machine data.

Subroutine call with M, T and D functions

For certain applications, it may be advantageous to replace M, T or D functions as well as a few NC language commands SPOS, SPOSA, by a subroutine call. This can be used, for example, to call the tool change routine.

Relevant machine data can be used to define and control subroutines having M, T or D functions. For example, for a gear stage change.

Program runtime/part counter

Information on the program runtime and the part count is provided to assist the machine tool operator.

The functions defined for this purpose are **not identical to the functions of tool management** and are intended primarily for systems without tool management.

8.2 Mode group

Mode group

A mode group combines NC channels with axes and spindles to form a machining unit.

A mode group contains the channels that are required to run simultaneously in the same mode from the point of view of the machining sequence.

The configuration of a mode group defines which channels are to be included in the group.

Note

This description assumes one mode group and one channel.

Functions that need several channels, e.g. "Axis replacement" function, are described in:
References:
Function Manual Extension functions; Mode Groups, Channels, Axis replacement (K5)

Assignment: Channel - Mode group

Axes and/or spindles are assigned to one channel.

The channel, in turn, is assigned to a mode group with the following machine data:

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP (channel valid in mode group)

If the same mode group is addressed in several channels, these channels together form a mode group.

Note

The control system does not recognize mode group-specific data. However, it is possible to make some channel-specific settings that pertain to the mode group.

Channel-specific assignments

Axes can be assigned to multiple channels that, in turn, are allocated to different mode groups. The axes can then be interchanged between these channels (axis replacement). Axis replacement functions independently of the mode group.

Machine axes and spindles are assigned to a channel. They differ as follows:

- Geometry axes can be operated in the path grouping.

  Using the master spindle, they can perform functions such as G96, G961, G331, G332, etc.
- Channel axes that are not defined as geometry axes can be moved as path axes, synchronous axes, positioning axes, PLC axes, and command axes.
- Special axes have no geometric relationship with one another.
- Master spindle geometry axes can perform functions using the master spindle.
- Auxiliary spindles are all other spindles/axes in the channel apart from the master spindle.

The GEOAX replacement command can be programmed to declare a channel axis as a geometry axis and define its geometry axis number. It is defined with SETMS, which spindle in the channel should be the master spindle.

Any axis in the channel can be configured as a spindle. The number of axes per channel depends on the controller version. In order to optimize the performance utilization, the available channel and axis configurations are limited depending on the hardware.

With SINUMERIK 840D sl, the following configurations are permissible depending on the HW/SW version:

- Up to 12 axes/spindles per channel
- Maximum of 31 axes or 20 spindles per NCU

For information about other axis configurations such as axis containers, link axes, reciprocating axes, main run, rotary, linear, master and slave axes and the various implementations, see SectionsK2: Axis Types, Coordinate Systems, Frames (Page 721) and S1: Spindles (Page 1383).

Mode group-specific interface signals

The exchange of mode group-specific signals to/from the mode group is transferred to DB11 in the user interface. In this way, the mode group can be monitored and controlled from the PLC or NCK.

The following table represents all the mode group-specific interface signals.

Signals from PLC to NCK
Mode group reset
Mode group stop axes plus spindles
Mode group stop
Mode change
Mode: JOG, MDA, AUTOM.
Single block: Type A, Type B
Machine function REF, REPOS, TEACH IN,

Signals from NCK to PLC
Mode strobe: JOG, MDA, AUTOMATIC
Machine function strobe: REF, REPOS, TEACH IN
All channels (1 to 10, max.) in Reset state
Mode group Ready
Active mode: JOG, MDA, AUTOMATIC
Digitizing
Active machine function: REF, REPOS, TEACH IN var. INC, 10000 INC ..... 1 INC

Change in mode group

A configuration change of a mode group with respect to its assigned channels requires a subsequent POWER ON.

The change is made using machine data:

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP.

Mode group numbers must be assigned contiguously starting with 1.

Machine data

There is no mode group-specific machine data.

Channel gaps

The channels to which a mode group is assigned with MD10010 are regarded as activated.

Instead of a mode group number, the number "0" can be assigned to channels. The result is as follows:

- The non-activated channel does not take up memory space in the controller.
- Series machines with similar designs can be kept uniform during configuration. Only the channels that can actually be used by the machine tool are activated (channels with mode group number greater than 0).

Special case:

Channel 1 must always be available!

⇒ If:
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP [0] = 0
is specified, the controller automatically sets
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP [0] = 1 (mode group 1).

Example configurations:

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[0] = 1

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[1] = 2

...

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[3] = 0 ; gap

...

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[8] = 1

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[9] = 2

8.2.1 Mode group stop

Function

The following NC/PLC interface signals are used to stop the traversing motions of the axes or of the axes and spindles in all mode group channels and to interrupt part program execution:

DB11 DBX0.5 (mode group stop)

DB11 DBX0.6 (mode group stop, axes plus spindles)

8.2.2 Mode group reset

Function

A mode group Reset is requested via a mode group-specific NC/PLC interface signal:

DB11 DBX0.7 = 1 (mode group reset)

Effect

Effect on the channels of mode group:

- Part program preparation (preprocessing) is stopped.
- All axes and spindles are decelerated to zero speed according to their acceleration curves without contour violation.
- Any auxiliary functions not yet output to the PLC are no longer output.
- The preprocessing pointers are set to the interruption point, and the block pointers are set to the beginning of the appropriate part programs.
- All initial settings (e.g. the G functions) are set to the parameterized values.
- All alarms with "Channel reset" criterion are canceled.

If all the channels of the mode group are in reset state, then:

- All alarms with "Mode group reset" cancel criterion are canceled.
- The NC/PLC interface indicates completion of the mode group Reset and the mode group's readiness to operate:

  DB11 DBX6.7 (all channels in the reset state)

  DB11 DBX6.3 = 1 (mode group ready)

8.3 Mode types and mode type change

Unique mode

All channels of a mode group are always in the same mode:

- AUTOMATIC
- JOG
- MDI

If individual channels are assigned to different mode groups, a channel switchover activates the corresponding mode group. This allows mode changes to be initiated via a channel switchover.

Modes

The following modes are available:

- **AUTOMATIC**

  Automatic execution of part programs:

  – Part program test.

  – All channels of the mode group can be active at the same time.

- **JOG in Automatic**

  JOG in AUTOMATIC is an extension of AUTOMATIC mode intended to simplify use. JOG can be executed without leaving AUTOMATIC mode if boundary conditions so permit.

- **JOG**

  Manual traversing of axes via traversing keys of the machine control panel or via a handwheel connected to the machine control panel:

  – Channel-specific signals and interlocks are taken into account for motions executed by means of an ASUB or via static synchronized actions.

  – Couplings are taken into account.

  – Every channel in the mode group can be active.

- **MDI**

  Manual Data Automatic (the blocks are entered via the user interface)

  – Restricted execution of part programs and part program sections.

  – Part program test.

  – A maximum of one channel per mode group can be active (applies only to TEACH IN).

  – Axes can only be traversed manually in subordinate machine functions such as JOG, REPOS or TEACH IN.

Applies to all modes

Cross-mode synchronized actions

Modal synchronized actions can be executed by means of IDS in all modes for the following functions parallel to the channel:

- Command axis functions
- Spindle functions
- Technology cycles

Selection

The user can select the desired mode by means of softkeys on the user interface.

This selection (AUTOMATIC, MDI, or JOG) is forwarded to the PLC on the NC/PLC interface, but is not activated:
DB11 DBX4.0, 0.1, 0.2 (strobe mode)

Activation and priorities

The mode of the mode group is activated via the NC/PLC interface:

DB11 DBX0.0, 0.1, 0.2 (mode)

If several modes are selected at the same time, the following priority applies:

Priority	Mode	Mode-group signal (NCK → PLC)
1st priority, high	JOG	DB11 DBX0.2
2nd priority, medium	MDI	DB11 DBX0.1
3rd priority, low	AUTOMATIC	DB11 DBX0.0

Display

The current mode of the mode group is displayed via the NC/PLC interface:

DB11 DBX6.0, 0.1, 0.2 (active mode)

Mode-group signal (NCK → PLC)	Active operating mode
DB11 DBX6.2	JOG
DB11 DBX6.1	MDI
DB11 DBX6.0	AUTOMATIC

Machine functions

Machine functions can be selected within a mode that also apply within the mode group:

- Machine functions within the **JOG** mode
  - REF (reference point approach)
  - REPOS (repositioning)
  - JOG retract (retraction motion in the tool direction)
- Machine functions within the **MDI** mode
  - REF (reference point approach)
  - REPOS (repositioning)
  - TEACHIN (teach-in of axis positions)

NC/PLC interface signals

- DB11 DBX5.0, 0.1, 0.2 (machine function strobe): Requirement
- DB11 DBX1.0, 0.1, 0.2 (machine function): Activation
- DB11 DBX7.0-2 (active machine function): Feedback signal

Channel states

- **Channel reset**

  The machine is in its initial state. This is defined by the machine manufacturer's PLC program, e.g. after POWER ON or at the end of the program.
- **Channel active**

  A program has been started, and the program is being executed or a reference point approach is in progress.
- **Channel interrupted**

  The running program or reference point approach has been interrupted.

Functions within modes

Modes are supplemented through user-specific functions. The individual functions are technology and machine-independent and can be started and/or executed from the three channel states "Channel reset", "Channel active" or "Channel interrupted".

Supplementary condition for submode TEACH IN

TEACH IN is not permissible for leading or following axes of an active axis grouping, e.g. for:

- Gantry-axis grouping or a gantry axis pair
- Coupled-axis grouping of master and slave axis

JOG in Automatic

JOG in AUTOMATIC mode is permitted if the mode group is in "RESET" state and the axis is jog-capable. "RESET" state for the mode group means:

- All channels are in the "RESET" state
- All programs are canceled
- DRF is not active in any channel

An axis is **JOG-capable** if it is not in any of the following states:

- PLC axis as concurrent positioning axis (request of the axis from the PLC)
- Command axis (the axis has been programmed by a synchronized action and the motion has not been completed yet)
- Rotating spindle (spindle rotating despite RESET)
- An asynchronous reciprocating axis

Note: The “jog-capable” property is independent of the “JOG in AUTOMATIC” function.

Activation

The function "JOG in AUTOMATIC" can be activated with the machine data:

MD10735 $MN_JOG_MODE_MASK

- Before POWER ON, the following machine data must be set:

  MD10735 $MN_JOG_MODE_MASK, bit 0 = 1

- The user switches to AUTO (PLC user interface DB11 DBX0.0 = 0→1 edge). “JOG in AUTOMATIC” is then active if the NCK previously had channel state “RESET” and program state “Aborted” in all mode group channels. The axis in question must also be “jog-capable”. DRF must be deactivated (if not already deactivated).
- RESET is initiated or the running program is finished with `M30`/`M2` in all mode group channels that do not have channel state “Reset” and program state “Aborted”.
- The relevant axis is automatically made "JOG-capable" (e.g. axis interchanget: PLC → NC).

Note: In most applications, the axes to be traversed are "JOG-capable" and with the switchover to AUTOMATIC, "JOG in AUTOMATIC" is also active.

Characteristics

- The +/– keys cause a JOG movement, and the mode group is switched **internally** to JOG. (i.e. “Internal JOG”).
- Moving the handwheels causes a JOG movement, and the mode group is switched **internally** to JOG, unless DRF is active.
- An ongoing JOG movement is not complete until the end position of the increment has been reached (if this has been set) or the movement has been aborted with “Delete distance-to-go”.

  In this way an increment can be stopped using Stop and then moved to the end using Start. The NCK remains in “Internal JOG” during this time. A partial increment is possible, but it must not be interrupted using Stop. There is a mode in which releasing the travel key causes interruption within an increment.
- Without any JOG movement, “JOG in AUTOMATIC” responds in the same way as “Automatic”. In particular, the Start key starts the selected part program and the appropriate HMI softkey initiates a block search.
- If JOG movement is active, the NCK is internally in JOG mode, and, thus, a block search request is refused and a Start cannot start the part program. Start starts any remaining increment or has no effect.
- While a mode group axis is being traversed in JOG mode, the mode group remains internally in JOG mode.
  Comment: This phase can begin with the JOG movement of an axis and end with the end of the JOG movement of *another* axis.
- Axis interchange is not possible for an axis with active JOG movement (the axis might change mode group). The NCK blocks any axis interchange attempt.
- The PLC user interface indicates "Automatic" mode:
  - DB11 DBX6.0, 6.1, 6.2 = 1
  - DB11 DBX7.0, 7.1, 7.2 = 0
- In “JOG in AUTOMATIC”, the NC/PLC interface displays whether the mode group is in “Mode group RESET”.
  - DB11 DBX6.4 (mode group has been reset, mode group 1)
  - DB11 DBX26.4 (mode group has been reset, mode group 2)
  - DB11 DBX46.4 (mode group has been reset, mode group 3)
- In “JOG in AUTOMATIC”, the NC/PLC interface displays whether the NC has automatically switched to “Internal JOG”.
  - DB11 DBX6.5 (NCK internal JOG active, mode group 1)
  - DB11 DBX26.5 (NCK internal JOG active, mode group 2)
  - DB11 DBX46.5 (NCK internal JOG active, mode group 3)

Supplementary conditions

"JOG in AUTOMATIC" can only switch internally to JOG if the mode group is in "Mode group RESET" state, i.e. it is not possible to jog immediately in the middle of a stopped program. The user can jog in this situation by pressing the JOG key or the Reset key *in all channels* of the mode group.

Selecting AUTOMATIC disables the INC keys and the user can/must press the INC keys again to select the desired increment. If the NCK switches to "Internal JOG", the selected increment is retained.

If the user attempts to jog the geometry or orientation axes, the NCK switches to "Internal JOG" and the movement executed. Several axes can be physically moved in this way; they must all be "JOG-capable".

Following the JOG movement, the NCK deactivates "Internal JOG" again and selects AUTO mode again. The internal mode change is delayed until the movement is complete. This avoids unnecessary multiple switching operations, e.g. when using the handwheel. The PLC can only rely on the "Internal JOG active" PLC signal.

The NCK will then switch to "Internal JOG" if the axis is not enabled.

See also

R1: Referencing (Page 1319)

8.3.1 Monitoring functions and interlocks of the individual modes

Channel status determines monitoring functions

Monitoring functions in operating modes

Different monitoring functions are active in individual operating modes. These monitoring functions are not related to any particular technology or machine.

In a particular mode only some of the monitoring functions are active depending on the operating status. The channel status determines which monitoring functions are active in which mode and and in which operating state.

Interlocking functions in operating modes

Different interlocks can be active in the different operating modes. These interlocking functions are not related to any particular technology or machine.

Almost all the interlocks can be activated in every mode, depending on the operating status.

8.3.2 Mode change

Introduction

A mode change is requested and activated via the mode group interface (DB11, ...). A mode group will either be in AUTOMATIC, JOG, or MDA mode, i.e. it is not possible for several channels of a mode group to take on different modes at the same time.

What mode transitions are possible and how these are executed can be configured in the PLC program on a machine-specific basis.

Note

The mode is not changed internally until the signal "Channel status active" is no longer pending. For error-free mode change however, all channels must assume a permissible operating mode.

Possible mode changes

The following table shows possible mode changes for one channel.

	AUTOMATIC		JOG				MDA		
				AUTO	MDA		JOG without handwheel		AUTO
	Reset	Interrupt	Reset	Interrupt	Interrupt	Reset	Interrupt	active	Interrupt
AUTOMATIC			X	X		X			
JOG	X	X				X	X		X
MDA	X	X	X		X				

Possible mode changes are shown by an "X".

Special cases

- **Errors during mode change**

  If a mode change request is rejected by the system, the error message "Operating mode cannot be changed until after NC Stop" is output. This error message can be cleared without changing the channel status.

- **Mode change disable**

  A mode change can be prevented by means of interface signal:
  DB11, DBX0.4 (Mode change disable).

  This suppresses the mode change request.

  The user must configure a message to the operator indicating that mode change is disabled. No signal is set by the system.

- **Mode change from MDA to JOG**

  If all channels of the mode group are in Reset state after a mode change from MDA to JOG, the NC switches from JOG to AUTO. In this state, part program commands `START` or `INIT` can be executed.

  If a channel of the mode group is no longer in Reset state after a mode change, the part program command `START` is rejected and Alarm 16952 is issued.

8.4 Channel

Assignment part program - channel

Part programs are assigned to channels.

Part programs of different channels are largely independent of each other.

Channel properties

A channel constitutes an "NC" in which one part program can be executed at a time. Machine axes, geometry axes and positioning axes are assigned to the channels according to the machine configuration and the current program status (AXIS CHANGE, GEO AXIS CHANGE, SETMS).

The system assigns each channel its own path interpolator with associated programming. Each channel can run its own machining program which is controlled from the PLC.

The following channel-specific functions make it possible for the channels to process part programs independently:

- Each channel has its own NC Start, NC Stop, RESET.
- One feedrate override and one rapid traverse override per channel.
- Dedicated Interpreter for each channel.
- Dedicated path interpolator for each channel which calculates the path points such that all the machining axes of the channel are controlled simultaneously from path axes.
- Selection and deselection of tool cutting edges and their length and radius compensations for a tool in a specific channel.

  For further information on the tool offset, see Section "W1: Tool offset (Page 1567)".
- Channel-specific frames and frames active in the channel for transforming closed calculation rules into Cartesian coordinate systems. Offsets, rotations, scalings, and mirrorings for geometry axes and special axes are programmed in a frame.

  For further information on frames, see Section "Zero offset external via system frames (Page 840)".
- Display of channel-specific alarm responses.
- Display of current machining sequence (axis position, current G functions, current auxiliary functions, current program block) for each channel.
- Separate program control functions for each channel.

These functions (with the exception of the display functions) are controlled and checked by the PLC with interface signals.

Channels in the same mode group always have to be operated in the same mode (AUTOMATIC, JOG, MDA).

Channel configuration

Channels can be filled with their own channel name via the following machine data:

MD20000 $MC_CHAN_NAME (channel name)

The various axes are then assigned to the available channels via machine data. There can be only one setpoint-issuing channel at a time for an axis/spindle. The axis/spindle actual value can be read by several channels at the same time. The axis/spindle must be registered with the relevant channel.

The following channel-specific settings can also be made using machine data:

- Position of deletions or the basic program settings of G groups via the machine data:

  MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)
- Auxiliary function groups regarding the combination and the output time.
- Transformation conditions between machine axes and geometry axes.
- Other settings for the execution of a part program.

Change in the channel assignment

An online change in the channel configuration cannot be programmed in a part program or PLC user program. Changes in the configuration must be made via the machine data. The changes become effective only after a new POWER ON.

Container axes and link axes

An axis container combines a group of axes in a container. These axes are referred to as container axes. This involves assigning a pointer to a container slot (ring buffer location within the relevant container) to a channel axis. One of the axes in the container is located temporarily in this slot.

Each machine axis in the axis container must be assigned at all times to exactly **one** channel axis.

Link axes can be assigned permanently to one channel or dynamically (by means of an axis container switch) to several channels of the local NCU or the other NCU. A link axis is a non-local axis from the perspective of one of the channels belonging to the NCU to which the axis is not physically connected.

The assignment between the link axes and a channel is implemented as follows:

- For permanent assignment using machine data:

  Allow the direct logic machine axis image to show link axes.

- For dynamic assignment:

  Allow the axis container slot machine data to show link axes.

Further information on link axes and container axes can be found in:
References:
Function Manual, Extended Functions; Several Operator Panel Fronts and Multiple NCUs, Distributed Systems (B3)

Interface signals

The signals of the 1st channel are located in the NC/PLC interface in DB21, the signals from channel 2 are located in DB22. The channel or channels can be monitored and controlled from the PLC or NCK.

Channel-specific technology specification

The technology used can be specified for each channel:

MD27800 $MC_TECHNOLOGY_MODE

In the delivery state, machine data is active for milling as standard.

Spindle functions using a PLC

In addition to function block FC18, spindle functions can also be started and stopped via the axial NC/PLC interface signals in parallel to part programs that are running.

Requirements:

- Channel status: "Interrupted" or "RESET"
- Program status: "Interrupted" or "canceled"

The following functions can be controlled from the PLC via interface signals:

- Stop (corresponds to M5)
- Start with clockwise direction of rotation (corresponds to M3)
- Start with counter-clockwise direction of rotation (corresponds to M4)
- Select gear stage
- Positioning (corresponds to M19)

For several channels, the spindle started by the PLC is active in the channel to which it is assigned at the start.

For further information on the special spindle interface, see Section "S1: Spindles (Page 1383)".

PLC-controlled single-axis operations

An axis can also be controlled from the PLC instead of from a channel. For this purpose, the PLC requests the axis from the NC via the NC/PLC interface:

DB31, ... DBX28.7 = 1 (PLC controls axis)

The following functions can be controlled from the PLC:

- Cancel axis/spindle sequence (equivalent to delete distance-to-go)
- Stop or interrupt axis/spindle
- Resume axis/spindle operation (continue the motion sequence)
- Reset axis/spindle to the initial state

For further information on the channel-specific signal exchange (PLC → NCK), see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)".

The exact functionality of independent single-axis operations is described in:

References:
Function Manual, Extended Functions; Positioning Axes (P2)

8.4.1 Global start disable for channel

User/PLC

A global Start disable can be set for the selected channel via the HMI or from the PLC.

Function

When Start disable is set, no new program starts are accepted for the selected channel. Start attempts are counted internally.

If a start is executed by the PLC before a global block disable is sent from the HMI to the NCK, the program is not stopped by the Start disable and its status is transmitted to the HMI.

NC Start disable and global Start disable have the same effect on the internal counter for starts that have been sent but not executed. (OPI variable startRejectCount).

Bypassing global Start disable

The interface signal:
DB21, ... DBX7.5 (PLC → NCK)
allows the PLC to temporarily bypass a global Start disable.

0: Global Start disable is effective
1: Global Start disable is temporarily canceled.

Messages

If desired, a message can be issued when a Start attempt occurs while a global block disable is active.

The control is exercised using machine data:

MD11411 $MN_ENABLE_ALARM_MASK Bit 6

1: Alarm 16956 appears: Channel %1, Program %2 cannot be started because of "Global Start disable".
0: Start attempts when a global block disable is set are not signaled by an alarm.

8.5 Program test

Several control functions are available for testing a new part program. These functions are provided to reduce danger at the machine and time required for the test phase. Several program functions can be activated at the same time to achieve a better result.

Test options

The following test options are described below:

- Program execution without setpoint outputs
- Program execution in singleblock mode
- Program execution with dry run feedrate
- Skip part program blocks
- Block search with or without calculation.

8.5.1 Program execution without setpoint outputs

Function

In the "Program test" status, a part program is executed without the output of axis or spindle setpoints.

The user can use this to check the programmed axis positions and auxiliary function outputs of a part program. This program simulation can also be used as an extended syntax check.

Selection

This function is selected via the operator interface in the "Program control" menu.

The selection sets the following interface signal:

DB21, ... DBX25.7 (program test selected)

This does not activate the function.

Activation

The function is activated via interface signal:

DB21, ... DBX1.7 (activate program test)

Display

The corresponding field on the operator interface is reversed and the interface signal in the PLC as a checkback of the active program test:

DB21, ... DBX33.7 (program test active)

Program start and program run

When the program test function is active, the part program can be started and executed (incl. auxiliary function outputs, wait times, G function outputs etc.) via the interface signal:

DB21, ... DBX7.1 (NC-Start)

The safety functions such as software limit switch, working area limits continue to be valid.

The only difference compared to normal program operation is that an internal **axis disable** is set for all axes (including spindles). The machine axes do not move, the actual values are generated internally from the setpoints that are not output. The programmed velocities remain unchanged. This means that the position and velocity information on the operator interface is exactly the same as that output during normal part program execution. The position control is not interrupted when this function is active, so the axes do not have to be referenced when the function is switched off.

⚠ CAUTION

The signals for exact stop:
DB31, ... DBX60.6/60.7 (exact stop coarse/fine)
mirror the actual status on the machine.

They are only canceled during program testing if the axis is pushed out of its set position (the set position remains constant during program testing).

With signal:
DB21, ... DBX33.7 (program test active)
both the PLC program and the part program can use variable $P_ISTEST to decide how to react or branch in response to these signals during testing.

Note

Dry run feedrate

"Program execution without axis motion" can also be activated with the function "Dry run feedrate". With this function, part program sections with a small programmed feedrate can be processed in a shorter time.

Note

Tool management

Because of the axis disable, the assignment of a tool magazine is not changed during program testing. A PLC application must be used to ensure that the integrity of the data in the tool management system and the magazine is not corrupted. The toolbox diskettes contain an example of the basic PLC program.

8.5.2 Program execution in single-block mode

Function

In case of "Program execution in single-block mode" the part program execution stops after every program block. If tool cutter radius compensation or a tool nose radius correction is selected, processing stops after every intermediate block inserted by the controller.

The program status switches to "Program status stopped".

The channel status remains active.

The next part program block is processed on NC Start.

Application

The user can execute a part program block-by-block to check the individual machining steps. Once the user decides that an executed part program block is functioning correctly, he can call the next block.

Single-block types

The following different types of single block are provided:

- Decoding single block

  With this type of single block, all blocks of the part program (even the pure computation blocks without traversing motions) are processed sequentially by "NC Start".

- Action single block (initial setting)

  With this type of single block, the blocks that initiate actions (traversing motions, auxiliary function outputs, etc.) are processed individually.

  Blocks that were generated additionally during decoding (e.g. for cutter radius compensation at acute angles) are also processed individually in singleblock mode.

  Processing is however not stopped at calculation blocks as these do not trigger actions.

The single-block types are determined via the user interface in the menu "Program controls".

⚠ CAUTION

In a series of G33/G34/G35 blocks, a single block is only operative if "dry run feed" is selected.

Calculation blocks are not processed in single-step mode (only if single decoding block is active).

SBL2 is also ineffective with G33/G34/G35.

Selection

It is possible to select the single-block mode:

- Via the machine control panel (key "Single Block")
- Via the user interface

  For an exact procedure, see:

  References:
  Operating Manual of the installed HMI application

Activation

The function is activated through the PLC basic program via the interface signal:

DB21, ... DBX0.4 (activate single block)

Display

Active single-block mode is indicated by a reversal in the relevant field in the status line on the user interface.

Because of the single-block mode, as soon as the part program processing has processed a part program block, the following interface signal is set:

DB21, ... DBX35.3 (program status interrupted)

Processing without single-block stop

Despite the selected single-block mode, a processing without the single-block stop can be set for specific program runs, e.g. for:

- Internal ASUBs
- User ASUBs
- Intermediate blocks
- Block search group blocks (action blocks)
- Init blocks
- Subprograms with `DISPLOF`
- Non-reorganizable blocks
- Non-repositionable blocks
- Reposition block without travel information
- Tool approach block.

The setting is made via the following machine data:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop)

References:
List Manual, Detailed Description of the Machine Data

8.5.3 Program execution with dry run feedrate

Function

During "Program execution with dry run feedrate" the traversing speeds which have been programmed together with G01, G02, G03, G33, G34 and G35, are replaced by a parameterized feedrate value:

SD42100 $SC_DRY_RUN_FEED (dry run feedrate)

The dry run feedrate also replaces the programmed revolutional feedrate in program blocks with G95.

The exact way of working of the parameterized dry run feedrate (SD42100) depends on the setting of another setting data:

SD42101 $SC_DRY_RUN_FEED_MODE (mode for test run speed)

Value	Meaning
0	Dry run feedrate is the maximum of the programmed feedrate and setting data SD42100. (Default setting!) ⇒ SD42100 becomes effective only when the stored value is greater than the programmed feedrate.
1	Dry run feedrate is the minimum of the programmed feedrate and SD42100. ⇒ SD42100 becomes effective only when the stored value is less than the programmed feedrate.
2	The value in SD42100 acts as the dry run feedrate, regardless of the programmed feedrate.
10	As in the case of "0", except thread cutting (G33, G34, G35) and thread boring (G331, G332, G63). These functions are executed as programmed.
11	As in the case of "1", except thread cutting (G33, G34, G35) and thread boring (G331, G332, G63). These functions are executed as programmed.
12	As in the case of "2", except thread cutting (G33, G34, G35) and thread boring (G331, G332, G63). These functions are executed as programmed.

A dry run feedrate can be selected in the automatic modes and activated on interruption of an automatic mode or end of a block.

For further information on influencing the feedrate, see Section "V1: Feedrates (Page 1507)".

Application

⚠ DANGER
Workpieces may not be machined when "dry run feedrate" is active because the altered feedrates might cause the permissible tool cutting rates to be exceeded and the workpiece or machine tool could be damaged.

Selection

This function is selected via the user interface in the "Program control" menu.

The selection sets the following interface signal:

DB21, ... DBX24.6 (dry run feedrate selected)

This does not activate the function.

Activation

The function is activated via interface signal:

DB21, ... DBX0.6 (activate dry run feed)

Display

Active dry run feedrate mode is indicated by a reversal in the relevant field in the status line on the user interface.

8.5.4 Skip part-program blocks

Function

When testing or breaking in new programs, it is useful to be able to disable or skip certain part program blocks during program execution. For this, the respective records must be marked with a slash.

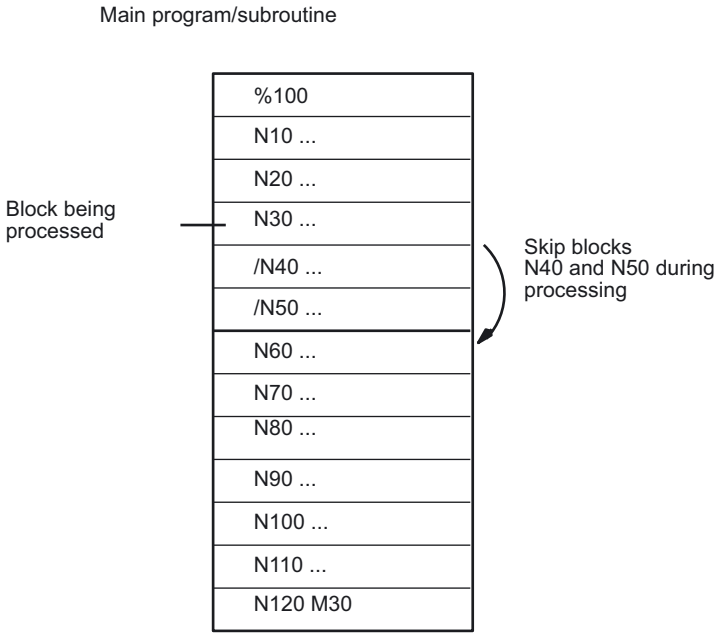


Figure 8-1 Skipping part program blocks

Selection

This function is selected via the operator interface in the "Program control" menu.

The selection sets the following interface signal:

DB21, ... DBX26.0 (skip block selected)

This does not activate the function.

Activation

The function is activated via the interface signal:

DB21, ... DBX2.0 (activate skip block)

Note

The "Skip part programs" function remains active during block searches.

Display

Activated "Skip block" function is indicated by a reversal of the relevant field on the operator interface.

8.6 Workpiece simulation

Function

The actual part program is completely calculated in the tool simulation and the result is graphically displayed in the user interface. The result of programming is verified without traversing the machine axes. Incorrectly programmed machining steps are detected at an early stage and incorrect machining on the workpiece prevented.

Simulation NCK

The simulation uses its own NCK instance (simulation NCK). Therefore, before a simulation is started, the real NCK must be aligned to the simulation NCK. With this alignment, all active machine data are read out of the NCK and read into the simulation NCK. The NCK and cycle machine data are included in the active machine data.

Compile cycles in simulation (only 840D sl)

Up to SW 4.4, no compile cycles are supported, from SW 4.4 and higher only selected compile cycles (CC) are supported for the workpiece simulation. The machine data of the supported compile cycles are aligned once after the control has powered-up. An alignment with "simulation start" does not take place!

Note

In part programs, CC-specific language commands and machine data of unsupported CCs **cannot** be used (see also paragraph "CC-commands in the part program").

Special motion of supported CCs (OEM transformations) are - under certain circumstances - incorrectly displayed.

CC-commands in the part program

Language commands in the part program of compile cycles that are not supported (OMA1 ... OMA5, OEMIPO1/2, G810 ... G829, own procedures and functions) therefore result in an alarm message and cancellation of the simulation without any individual handling.

Solution: Individually handle the missing CC-specific language elements in the part program ($P_SIM query). Example:

```
Program code                Comment
N1 G01 X200 F500
IF (1==$P_SIM)
N5 X300                     ; CC not active for simulation.
ELSE
N5 X300 OMA1=10
ENDIF
```

8.7 Block search

Function

Block search offers the possibility of starting part program execution from almost any part program block.

This involves the NC rapidly performing an internal run through the part program (without traversing motions) to the selected target block during block search. Here, every effort is made to achieve the exact same control status as would result at the target block during normal part program execution (e.g. with respect to axis positions, spindle speeds, loaded tools, NC/PLC interface signals, variable values) in order to be able to resume automatic part program execution from the target block with minimum manual intervention.

Block search types

- **Type 1: Block search without calculation**

  Block search without calculation is used to find a part program block in the quickest possible way. No calculation of any type is performed. The control status at the target block remains unchanged compared to the status before the start of the block search.

- **Type 2: Block search with calculation at contour**

  Block search with calculation at contour is used to enable the programmed contour to be approached in any situation. On NC Start, the start position of the target block or the end position of the block before the target block is approached. This is traversed up to the end position. Processing is true to contour.

- **Type 4: Block search with calculation at block end point**

  Block search with calculation at block end point is used to enable a target position (e.g. tool change position) to be approached in any situation. The end position of the target block or the next programmed position is approached using the type of interpolation valid in the target block. This is not true to contour.

  Only the axes programmed in the target block are moved. If necessary, a collision-free initial situation must be created manually on the machine in "JOG REPOS" mode before the start of further automatic part program execution.

- **Type 5: Block search with calculation in "Program test" (SERUPRO) mode**

  SERUPRO (**se**arch **ru**n by **pro**gramtest) is a cross-channel block search with calculation. Here, the NC starts the selected part program in "Program test" mode. On reaching the target block, the program test is automatically deselected. This type of block search also enables interactions between the channel in which the block search is being performed and synchronized actions as well as with other NC channels.

Note

For further explanations regarding the block search, see Section "Behavior during block search (Page 447)".

Subsequent actions

After completion of a block search, the following subsequent actions may occur:

- Type 1 - Type 5: Automatic Start of an ASUB

  When the last action block is activated, a user program can be started as an ASUB.

- Type 1 - Type 4: Cascaded block search

  A further block search with a different target specification can be started from "Search target found".

8.7.1 Sequence for block search of the type 1, 2 and 4

Time sequence

The block search (Types 1, 2, and 4) proceeds as follows:

1. Activation via the user interface
2. Search target found, or alarm if target cannot be found
3. NC Start for output of action blocks
4. NC Start for program continuation.

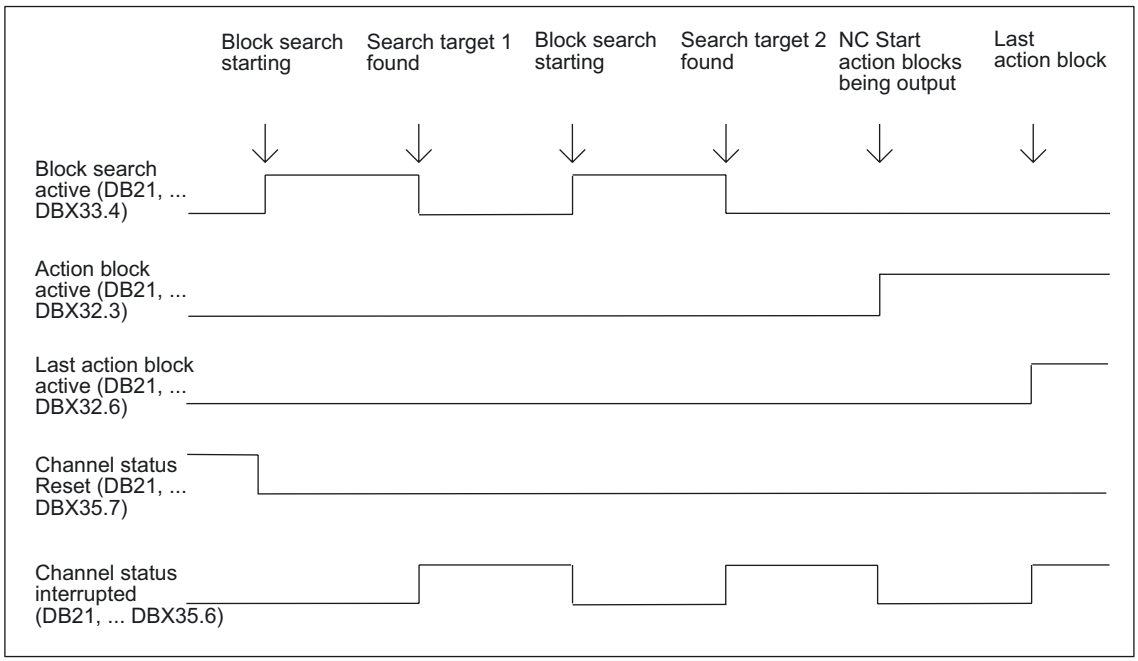


Figure 8-2 Time sequence of interface signals

Interface signals

In the PLC, the following interface signals are set according to the time sequence shown in the figure:

DB21, ... DBX33.4 (block search active)

DB21, ... DBX32.3 (action block active)

DB21, ... DBX32.4 (approach block active)

DB21, ... DBX32.6 (last action block active)

DB21, ... DBX1.6 (PLC action complete)

Continuation mode after block search

Following the block search, the program can be resumed via interface signal:
DB21, ... DBX7.1 (NC Start).

If an axis is first programmed after "Block search with calculation at block end point", the incremental value can be added to the value accumulated up to the search target using setting data
SD42444 $SC_TARGET_BLOCK_INCR_PROG.

Action blocks

Action blocks contain the actions accumulated during "Block search with calculation", such as auxiliary function outputs and tool (T, D), spindle (S), and feedrate programming commands. During "block search with calculation" (contour or block end point), actions such as M function outputs are accumulated in so-called action blocks. These blocks are output on an NC Start after "Search target found".

Note

With the action blocks, the accumulated spindle programming (S value, M3/M4/M5/M19, SPOS) also becomes active.

The PLC user program must ensure that the tool can be operated and that, if necessary, the spindle programming is reset via PLC signal:
DB31, ... DBX2.2 (spindle reset)
or the spindle programming is not output.

Single-block processing: MD10702 $MN_IGNORE_SINGLEBLOCK_MASK ()

By setting bit 3 = 1, it is possible to prevent a stop after every action block in single-block mode.

Supplementary conditions for approach block / target block

Block search type 2

Interface signal:
DB21, ... DBX32.4 (approach block active)
is **only** set with "Block search with calculation at contour" because a separate approach block is not generated with "Block search with calculation at block end point" (the approach block is the same as the target block).

Block search type 4

The approach motion "Search with calculation to block end point" is performed using the type of interpolation valid in the target block. This should be G0 or G1, as appropriate. With other types of interpolation, the approach motion can be aborted with an alarm (e.g. circle end point error on G2/G3).

8.7.2 Block search in connection with other NCK functions

8.7.2.1 ASUB after and during block search

Synchronization of the channel axes

With the start of an ASUB after "block search with calculation", the actual positions of all channel axes are synchronized during preprocessing.

Effects:

- System variable: $P_EP (programmed end position)

  In the ASUB, the system variable provides: $P_EP (programmed end position) the current actual position of a channel axes in the work piece coordinate system.

  $P_EP == "current actual position of the channel axis"

- System variable: $AC_RETPOINT (repositioning point in the ASUB)

  In the ASUB, the system variable provides: $AC_RETPOINT (repositioning point in the ASUB) the actual position of a channel axis in the workpiece coordinate system accumulated with a block search.

  $AC_RETPOINT == "collected search position of the channel axis (WCS)"

Block search type 2

For block search type 2 (block search with calculation on contour) the following part program command must be programmed at the conclusion of the ASUB:

`REPOSA` (repositioning on the contour; linear; all channel axes)

Effect:

- All channel axes are moved to their search position that was collected during the block search.
- $P_EP == "accumulated search position of the channel axis (WCS)"

Block search type 4 and part program command `REPOS`

After block search type 4 (block search with calculation at block end point) no automatic repositioning is initiated during the following period of time by the part program command `REPOS`:

- Start: NC/PLC interface signals: DB21,... DBB32, Bit6 (last action block active) == 1
- End: Continuing the part program processing per NC START,.

The start point of the approach movement is represented by the current axis positions of the channel axes at the time of the NC start command. The end point results from the other transversing movements programmed in the part program.

For block search type 4, no approach movement is generated by the NC.

Effect:

- After exiting the ASUB, the system variable $P_EP thus provides the actual position, on which the channel axes of the ASUB were positioned (or manual (mode: JOG).

  $P_EP == "current actual position of the channel axis"

8.7.2.2 PLC actions after block search

To allow activation of PLC actions (starting of ASUBs, call-up of PLC functions) after the end of the block search at a defined point, there is the NCK/PLC interface signal:

DB21, ... DB32.6 (last action block active) == 1

This means that all action blocks are processed and that actions are possible by the PLC (ASUB, FC) or the operator (overstoring, mode change after JOG/REPOS). This allows the PLC to perform another tool change, for example, before the start of the transversing movement.

By default, alarm 10208 is output at this moment to notify the operator that another NC START is needed to continue program execution.

In combination with alarm 10208, the following interface signals are set:

DB21, ... DBX36.7 (NCK alarm with processing stop)

DB21, ... DBX36.6 (channel-specific NCK alarm is present)

PLC-controlled alarm triggering

The setting by which alarm 10208 is only triggered after ending the PLC action, is done via machine data:
MD11450 $MN_SEARCH_RUN_MODE, Bit 0 = 1

Bit	Value	Meaning
0	1	With the change of the last action block after a block search, the following takes place: • Execution of the part program is stopped • DB21, ... DBB32.6 (last action block active) = 1 • Alarm display: Alarm 10208 only if the following applies: DB21, ... DBX1.6 (PLC action ended) == 1

8.7.2.3 Spindle functions after block search

Control system response and output

The behavior with regard to the spindle functions after completion of the block search can be set via machine data:
MD11450 $MN_SEARCH_RUN_MODE, bit 2

Bit	Value	Meaning
2	0	Output of spindle auxiliary functions (M3, M4, M5, M19, M70) in action blocks.
	1	Output of the auxiliary functions is suppressed in the action blocks. The spindle programming that accumulated during the block search can be output at a later point in time (e.g. via ASUB). The program data for this is stored in the following system variables: • $P_SEARCH_S • $P_SEARCH_SDIR • $P_SEARCH_SGEAR • $P_SEARCH_SPOS • $P_SEARCH_SPOSMODE

System variables

The spindle-specific auxiliary functions are always stored in the following system variables on block search, irrespective of the programming described above:

System variable	Description
$P_SEARCH_S[n]	Collected spindle speed, value range = { 0 ... Smax }
$P_SEARCH_SDIR[n]	Collected spindle rotation direction, value range = { 3, 4, 5, -5, -19, 70 }
$P_SEARCH_SGEAR[n]	Collected spindle gear stage M function, value range = { 40 ... 45 }
$P_SEARCH_SPOS[n]	Collected spindle position, value range = { 0 ... MD30330 $MA_MODULO_RANGE } Collected traverse path, value range = { -100,000,000 ... 100,000,000 }
$P_SEARCH_SPOSMODE[n]	Collected position approach mode, value range = { 0 ... 5 }

For later output of the spindle-specific auxiliary functions, the system variables can be read, for example, in an ASUB, and then output after output of the action blocks:
DB21, ... DBX32.6 == 1 (last action block active)

Note

The contents of the system variables $P_S, $P_DIR and $P_SGEAR may be lost after block search due to synchronization operations.

For more detailed information on ASUB, block search and action blocks, see Sections "Output suppression of spindle-specific auxiliary functions (Page 453)" and "Program test (Page 511)".

8.7.2.4 Reading system variables for a block search

Different system variables are available in the NC language to access values of the NC areas preprocessing, main run, or servo/drive:

$P_...	System variables that start with $P supply the pre-processing state (i.e. the programmed values).
$A_...	System variables that start with $A supply the main run state.
$V_...	System variables that start with $V provide data that are received from the servo/drive.

During a type 2 and 4 block search, no blocks access the main run. Therefore, it should be ensured that system variables, which reflect the main run or servo/drive states, are not changed by the search. Where necessary, for this variable, the block search must be executed in a special fashion by querying the machining type with $P_SEARCH in the NC program.

System variables that start with $P can be simply used in all search types.

8.7.3 Automatic start of an ASUB after a block search

Parameter assignment

Making the function effective

The automatic ASUB start after a block search is activated by the following MD setting:

MD11450 $MN_SEARCH_RUN_MODE, bit 1 = 1

Program to be activated

In the default setting, the program **_N_PROG_EVENT_SPF** is activated from the directory _N_CMA_DIR as ASUB after the block search by changing the last action block. If another program is to be activated, then the name of this user program must be entered in the following machine data:

MD11620 $MN_PROG_EVENT_NAME

Behavior when the single-block processing is set

Via the following channel-specific machine data it can be set, whether the activated ASUB is processed without interruption despite a set single-block processing or whether the single-block processing is to be made active:

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK

Bit	Value	Meaning
4	0	Single-block processing is active.
	1	Single-block processing is suppressed.

Behavior when the read-in disable is set

Via the following channel-specific machine data it can be set, whether the activated ASUB is processed without interruption despite a set read-in disable (DB21, ... DBX6.1 = 1), or whether the read-in disable is to be made active:

MD20107 $MC_PROG_EVENT_IGN_INHIBIT

Bit	Value	Meaning
4	0	Read-in disable is active.
	1	Read-in disable is suppressed.

Note

For further information on the parameterization of MD11620, MD20108 and MD20107, see Section "Parameterization (Page 603)".

Programming

The event that has started this ASUB can be determined by scanning the system variable $P_PROG_EVENT. In case of an automatic activation after a block search $P_PROG_EVENT returns the value "5".

Sequence

Sequence of automatic start of an ASUB after a block search

1. Start block search (with/without calculation, at contour, at end-of-block point).
2. Stop after "Search target found".
3. NC Start for output of action blocks.
4. Last action block is activated.
5. Automatic start of /_N_CMA_DIR/_N_PROG_EVENT_SPF (default) as an ASUB.
6. The NC will stop after changing the last ASUB block (REPOSA command) and the following NC/PLC interface signal is set:

   DB21, ... DBX32.6 (last action block active)

   The alarm 10208 "Enter NC-start for program continuation" is generated.

Note

If bit 0 is set to "1" in MD11450 $MN_SEARCH_RUN_MODE, then the alarm 10208 is generated only when the PLC requests this by setting the following NC/PLC interface signal:

DB21, ... DBX1.6 (PLC action complete)

8.7.4 Cascaded block search

Functionality

The "Cascaded block search" function can be used to start another block search from the status "Search target found". The cascading can be continued after each located search target as often as you want and is applicable to the following block search functions:

- Type 1 block search without calculation
- Type 2 block search with calculation at contour
- Type 3 block search with calculation at block end point

Note

Another "cascaded block search" can be started from the stopped program execution only if the search target has been found.

Activation

The "cascaded block search" is configured in the existing machine data:
MD11450 $MN_SEARCH_RUN_MODE

- Cascaded block search is enabled (i.e., several search targets can be specified) with Bit 3 = 0 (FALSE).
- For compatibility reasons, the cascaded block search can be disabled with Bit 3 = 1 (TRUE). By default, the cascaded block search is set with Bit 3 = 0.

Execution behavior

Search target found, restart search

When the search target is reached, the program execution stops and the search target is displayed as a current block. After each located search target, a new block search can be repeated as often as you want.

Change search target specifications

You can change the search target specifications and block search function prior to each block search start.

Example: Sequence with cascaded block search

- RESET
- Block search up to search target 1
- Block search up to search target 2 → "Cascaded block search"
- NC Start for output of the action blocks → Alarm 10208
- NC Start → Continue program execution

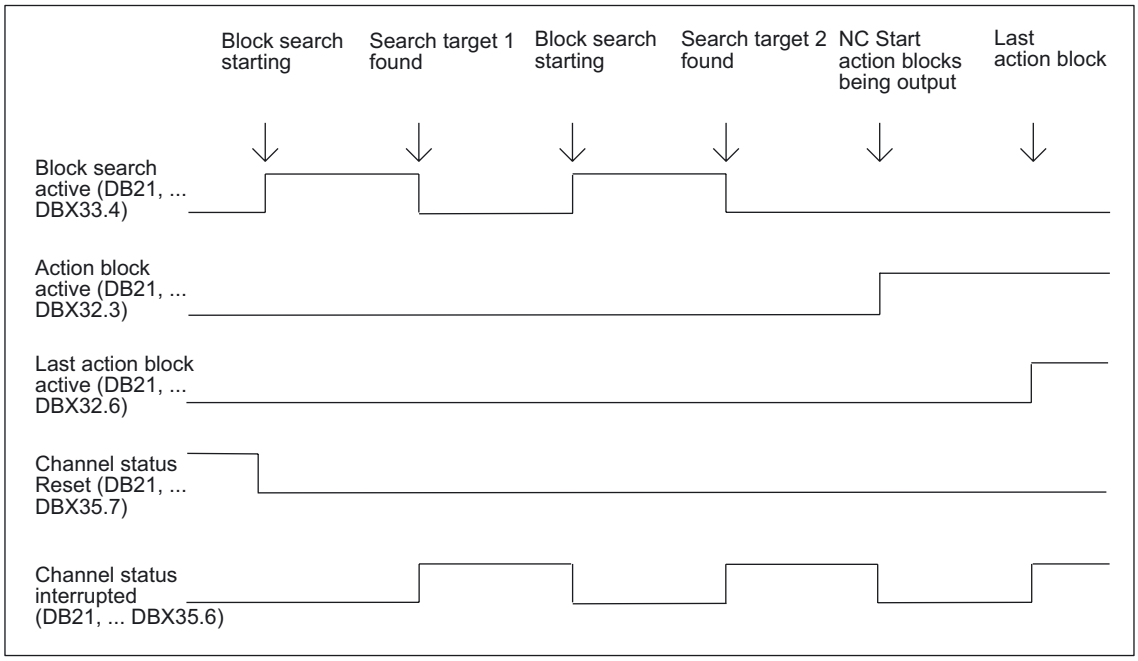


Figure 8-3 Chronological order of interface signals

8.7.5 Examples for block search with calculation

Selection

From the following examples, select the type of block search that corresponds to your task.

Type 4 block search with calculation at block end point

Example with automatic tool change after block search with active tool management:

1. Set machine data:

   MD11450 $MN_ SEARCH_RUN_MODE to 1

   MD11602 $MN_ASUB_START_MASK Bit 0 = 1 (ASUB Start from stopped state)

2. Select ASUB "BLOCK_SEARCH_END" from PLC via FB4 (see also Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)").
3. Load and select part program "WORKPIECE_1".
4. Search to block end point, block number `N220`.
5. HMI signals "Search target found".
6. NC Start for output of action blocks.
7. With PLC signal:
   DB21... DB32.6 (last action block active)
   the PLC starts ASUB "BLOCK_SEARCH_END" via FC9 (see also Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)").
8. After the end of the ASUB (can be evaluated, e.g. via M function `M90` to be defined, see example for block `N1110`), the PLC sets signal
   DB21, ... DBX1.6 (PLC action complete).

   Alternatively, NC/PLC interface signal:
   DB21-DB30 DBB318 bit 0 (ASUB is stopped)
   can be scanned.

   As a result, Alarm 10208 is displayed, i.e. other actions can now be performed by the operator.

9. Manual operator actions (JOG, JOG-REPOS, overstoring)
10. Continue part program with NC Start.

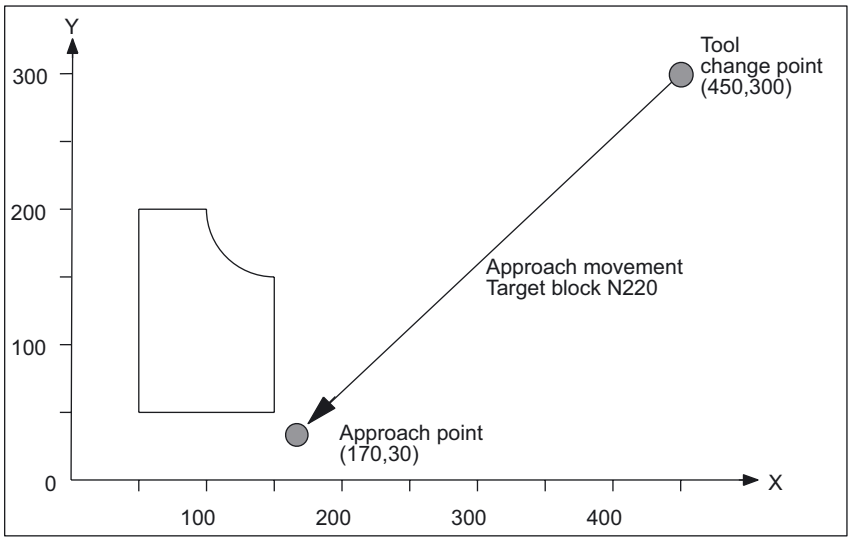


Figure 8-4 Approach motion for search to block end point (target block N220)

Note

"Search to contour" with target block N220 would generate an approach motion to the tool change point (start point of the target block).

Type 2 block search with calculation at contour

Example with automatic tool change after block search with active tool management:

1. to 3. Same as example for Type 4 block search
4. Search to contour, block number N260
5. to 10. Same as example for Type 4 block search

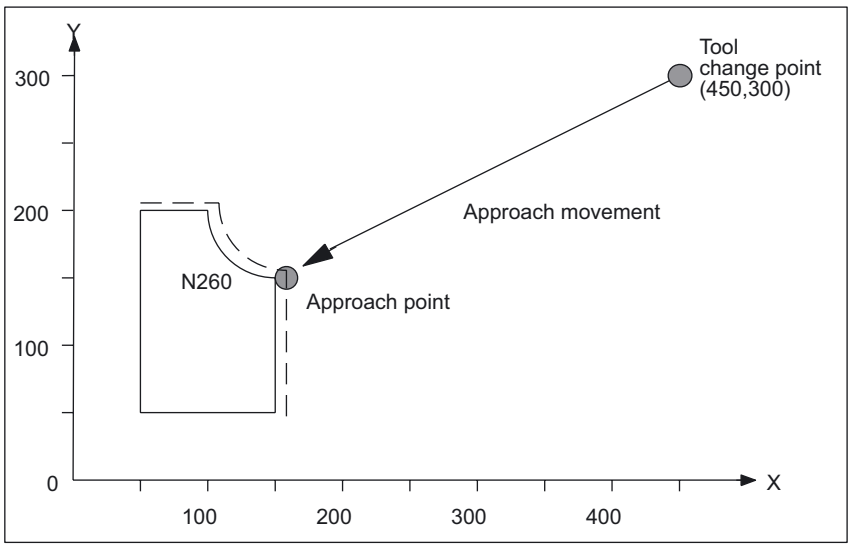


Figure 8-5 Approach motion for search to contour (target block N260)

Note

"Search to block end point" with target block N260 would result in Alarm 14040 (circle end point error).

Part programs for Type 4 and Type 2

PROC WORKPIECE_1

```
Program code                                    Comment
; Main program
...
;Machine contour section 1 with "CUTTER_1"tool
...
N100 G0 G40 X200 Y200                          ; Deselect radius compensation
N110 Z100 D0                                   ; Deselect length correction
;End of contour section 1
;
;Machine contour section 2 with "CUTTER_2"tool
N200 T="CUTTER_2"                              ; Preselect tool
N210 WZW                                       ; Call tool change routine
N220 G0 X170 Y30 Z10 S3000 M3 D1               ; Approach block for contour section 2
N230 Z-5                                       ; Infeed
N240 G1 G64 G42 F500 X150 Y50                  ; Start point of contour
N250 Y150
N260 G2 J50 X100 Y200
N270 G1 X50
```

```
Program code                                        Comment
N280 Y50
N290 X150
N300 G0 G40 G60 X170 Y30                    ; Deselect radius compensation
N310 Z100 D0                                ; Deselect length correction
End of contour section 2
...
M30
PROC WZW
;Tool change routine
N500 DEF INT TNR_AKTIV                      ; Variable for active T number
N510 DEF INT TNR_VORWAHL                    ; Variable for preselected T number
N520 TNR_AKTIV = $TC_MPP6[9998,1]           ; Read T number of active tool
N530 GETSELT(TNR_VORWAHL)                   ; Read T number of preselected tool
;
;Execute tool change only if tool is not yet active
N540 IF TNR_AKTIV == TNR_VORWAHL GOTOF
ENDE
N550 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0  ; Approach tool change position
N560 M6                                     ; Execute tool change
;
END: M17
PROC SUCHLAUF_ENDE SAVE
;ASUB for calling the tool change routine after block search
N1000 DEF INT TNR_AKTIV                     ; Variable for active T number
N1010 DEF INT TNR_VORWAHL                   ; Variable for preselected T number
N1020 DEF INT TNR_SUCHLAUF                  ; Variable for T number determined in
                                              search
N1030 TNR_AKTIV = $TC_MPP6[9998,1]          ; Read T number of active tool
N1040 TNR_SUCHLAUF = $P_TOOLNO              ; Read T number determined by search
N1050 GETSELT(TNR_VORWAHL)                  ; Read T number of preselected tool
N1060 IF TNR_AKTIV ==TNR_SUCHLAUF GOTOF ASUP_ENDE
N1070 T = $TC_TP2[TNR_SUCHLAUF]             ; T selection by tool name
N1080 WZW                                   ; Call tool change routine
N1090 IF TNR_VORWAHL == TNR_SUCHLAUF GOTOF ASUP_ENDE
N1100 T = $TC_TP2[TNR_VORWAHL]              ; Restore T preselection by tool name
ASUP_ENDE:
N1110 M90                                   ; Feedback to PLC
N1120 REPOSA                                ; ASUB end
```

8.8 Block search Type 5 SERUPRO

8.8.1 Description of the function

Block search type 5, block search with calculation in the "Program test" mode (SERUPRO, **"Se**arch-**Ru**n by **Pro**gram test") enables a cross-channel block search with calculation at a selectable interruption point. Taking into account existing program coordination commands, all the status data required to continue the program in the interrupted channels is determined during SERUPRO and then the NC and PLC set to a state permitting the program continuation.

Before repositioning with subsequent continuation of the program execution, all the output states that may still be required can be automatically generated via a user-specific ASUB.

Channels

In combination with the HMI, SERUPRO is provided for the following channels:

- For the current SERUPRO channel only (1)
- For all channels with the same workpiece name as the SERUPRO channel (2)
- For all channels with the same mode group as the SERUPRO channel (3)
- For all channels of the NCU (4)

The scope of channels for SERUPRO is selected by means of configuration file **maschine.ini**, in Section [BlockSearch]:

Section [BlockSearch]	Enable search function for HMI and select search configuration
SeruproEnabled=1	;SERUPRO softkey available for HMI. Default value is (1)
SeruproConfig=1	;Number (1) to (4) of above indicated channel grouping. Default value is (1)

All other channels started with SERUPRO are operated in "Self-Acting SERUPRO" mode. Only the channel in which a target block has been selected can be started with a block search in SERUPRO mode.

Activation

SERUPRO is activated via the HMI. SERUPRO is operated using the "Prog.Test Contour" softkey.

SERUPRO uses REPOS to approach the target block.

Chronological sequence of SERUPRO

1. Via HMI, softkey "Pog. test contour" and the search target are operated.
2. The NC now automatically starts the selected program in "Program test" mode.
   - In this mode, axes are not traversed.
   - Auxiliary functions $A_OUT and the direct PLC IO are output.
   - The auxiliary functions of the target block are not output.
3. The NC stops at the beginning of the target block, deselects the program test internally, and displays the Stop condition "Wait: Search target found".
4. If the user-specific ASUB "PROG_EVENT.SPF" is available, it is started automatically.
5. Repositioning is performed with the next NC start (REPOS).

   The REPOS operation is performed via a system ASUB and can be extended using the "Editable ASUB" function.

Boundary conditions for block search SERUPRO

The SERUPRO function may only be activated in "AUTOMATIC" mode and may only be aborted in program state (channel state RESET).

If in normal mode **only** the PLC starts commonly several channels, then this can be simulated by SERUPRO in each channel.

With machine data setting:
MD10708 $MN_SERUPRO_MASK, bit 1 = 0,
alarm 16942: "Channel %1 Start program command action %2<ALNX> not possible"
aborts the simulation if part program command `START` is used.

Machine data:
MD10707 $MN_PROG_TEST_MASK
allows shutdown in the stopped state and has no effect on the SERUPRO operation. The default setting allows program testing to be deactivated only in the RESET state.

Note

After program testing has been deactivated, a REPOS operation is initiated that is subject to the same restrictions as a SERUPRO approach operation. Any adverse effects can be inhibited using an ASUB.

Controlling SERUPRO behavior

For the functions listed below as an example, the SERUPRO behavior can be set specifically for the NC:

- Programmed stop (M0)
- Program coordination command START
- Group SERUPRO
- Cross-channel exiting of SERUPRO
- Override

MD10708 $MN_SERUPRO_MASK = <behavior with SERUPRO>

Channel-specific initial settings for SERUPRO

The channel-specific basic settings are normally specified with the following machine data after a part program start:

MD20112 $MC_START_MODE_MASK= <initial settings>

You can specify your own initial settings for SERUPRO which replace the initial settings from MD20112:

MD22620 $MN_START_MODE_MASK_PRT = <SERUPRO initial settings>

The SERUPRO initial settings must be explicitly released via:

MD22621 $MC_ENABLE_START_MODE_MASK_PRT = 1

NC/PLC interface signal "Block search via program test is active"

The block search via program test is displayed using the NC/PLC interface signal:
DB21, ... DBX318.1 == 1

The interface signal is set from the start of the block search until the target block is inserted into the main run.

For user-defined ASUB after the SERUPRO operation

Note

If the machine manufacturer decides to start an ASUB after the SERUPRO operation as described in point 7, the following must be observed:

Stopped status acc. to point 6:

Machine data:
MD11602 $MN_ASUP_START_MASK
and
MD11604 $MN_ASUP_START_PRIO_LEVEL
allow the NCK to start the ASUB from stopped status automatically via the FC9 block.

Acknowledgement of FC9 only after completion of REPOS block:

The ASUB can only be signaled as complete from the FC9 block with "ASUB Done" if the REPOS block has also been completed.

Deselection of assigned REPOS operation after point 8:

The start of the ASUB deselects the assigned REPOS operation!

Therefore, the ASUB should be ended with REPOSA in order to retain the REPOS operation.

Deleting an unwanted REPOS operation:

The unwanted REPOS operation is deleted by completing the ASUB with `M17` or `RET`.

Special handling of ASUB:

As a basic rule, an ASUB that ends with REPOS and is started from stopped state receives special treatment.

The ASUB stops automatically before the REPOS block and indicates this via:

DB21, ... DBX318.0 (ASUB is stopped)

Automatic ASUB start

The ASUB in path:
/_N_CMA_DIR/_N_PROG_EVENT_SPF
is started automatically in SERUPRO approach with machine data:
MD11450 $MN_SEARCH_RUN_MODE, bit 1 = 1
according to the following sequence:

1. The SERUPRO operation has been performed completely.
2. The user presses "NC start".
3. The ASUB is started.
4. The NC stops automatically **before** the `REPOS` part program command and the message "Press NC start to continue the program" appears.
5. The user presses "NC start" again.
6. The NC executes the repositioning motion and continues the part program at the target block.

Note

The automatic ASUB start with MD11450 requires **Starts** to continue the program.

The procedure is in this respect similar to other search types.

8.8.2 REPOS

MD11470

REPOS occurs according to machine data:

MD11470 $MN_REPOS_MODE_MASK

Case A:

The REPOS operation moves all axes from the current position to the start of the target block in a single block.

MD11470 $MN_REPOS_MODE_MASK Bit 3 = 1

Case B:

The path axes are repositioned together in one block. The SPOS and POS axes are repositioned in the residual block.

MD11470 $MN_REPOS_MODE_MASK Bit 3 = 0

8.8.2.1 Continue machining at the contour after SERUPRO search target found

User information regarding the REPOS operation

REPOS is generally used to interrupt an ongoing machining operation and to continue machining after the interruption.

In the SERUPRO approach, on the other hand, a program section must be "executed" later. This is the case, when SERUPRO has finished the simulation and is to be moved again to the target block. SERUPRO refers to the existing REPOS function, which the user can adapt as necessary.

SERUPRO approach

The user can change the REPOS behavior of individual axes at specific times to reposition certain axis types either earlier, later, or not at all. This affects SERUPRO approach in particular. Repositioning motions of some axes can also be controlled independently of SERUPRO approach during the REPOS operation.

⚠ CAUTION

The REPOS operation moves all axes from the current position to the start of the target block in a single block with the appropriate setting of machine data:
MD11470 $MN_REPOS_MODE_MASK Bit 3.

During this process, the NC cannot detect any possible collisions with the machine or the workpiece!

Protection zones and software limits are monitored.

Set REPOS response

Machine data:
MD11470 $MN_REPOS_MODE_MASK
can be used to set bits that specify the behavior of the NC during repositioning.

Bit 0 = 1	The dwell time is resumed at the point of interruption in the residual repositioning block.
Bit 1 = 1	Reserved
Bit 2 = 1	Prevent repositioning of individual axes using NC/PLC interface signals.
Bit 3 = 1	Reposition positioning axes in the approach block during block search via program test (SERUPRO).
Bit 4 = 1	Positioning axes in approach block on every REPOS.
Bit 5 = 1	Modified feedrates and spindle speeds are valid immediately in the residual block. Otherwise, not until the next block.
Bit 6 = 1	After SERUPRO, neutral axes and positioning spindles in the approach block are repositioned as command axis.
Bit 7 = 1	The level of interface signal: DB31, ... DBX10.0 (REPOSDELAY) is read if REPOSA is interpreted. Axes, which are neither geometry nor orientation axes, are then excluded by REPOS and are not moved.

Repositioning with controlled REPOS

At any point during processing, a part program can be interrupted and an ASUB started with a REPOS.

For path axes, the REPOS mode can be controlled by the PLC via NC/PLC interface signals to reposition on the contour. This mode is programmed in the part program and defines the approach behavior (see Section "Repositioning on contour with controlled REPOS (Page 551)").

The REPOS behavior of individual axes can also be controlled via NC/PLC interface signals and is enabled with machine data:
MD11470 $MN_REPOS_MODE_MASK BIT 2==1.

Path axes cannot be influenced individually. For all other axes that are not geometry axes, repositioning of individual axes can be prevented temporarily and also delayed. NC/PLC interface signals can be used to subsequently re-enable or to continue blocking individual channel axes that REPOS would like to traverse.

⚠DANGER

Signal:
DB31, ... DBX2.2 (delete distance-to-go, axis-specific)
produces the following dangerous behavior with ”Prevent repositioning of individual axes" via:
MD11470 $MN_REPOS_MODE_MASK (Bit 2==1).

As long as an axis is programmed incrementally after the interruption, the NC approaches different positions than those approached with no interruption (see example below).

Example: Axis is programmed incrementally

Axis A is positioned at 11° before the REPOS operation; the programmed operation in the interruption block (target block for SERUPRO) specifies 27°.

Any number of blocks later, this axis is programmed to move incrementally through 5° with:
`N1010 POS[A]=IC(5) FA[A]=1000.`

With interface signal:
DB31, ... DBX10.0 (REPOSDELAY)
the axis does not traverse in the REPOS operation and is moved to 32° with `N1010`.

(The user may have to deliberately acknowledge the travel from 11° to 27°.)

Caution:
The axis is programmed incrementally after the interruption.

In the example, the NC moves to 16° (instead of 32°).

A) Start axes individually

The REPOS behavior for SERUPRO approach with several axes is selected with:

MD11470 $MN_REPOS_MODE_MASK BIT 3 == 1

The NC commences SERUPRO approach with a block that moves **all** positioning axes to the programmed end and the path axis to the target block.

The user starts the individual axes by selecting the appropriate feedrate enables. The target block motion is then executed.

B) Reposition positioning axes in the repositioning block

Positioning axes are not repositioned in the residual block but rather in the repositioning block, and their effect is not limited to the block search via program test on SERUPRO approach.

MD11470 $MN_REPOS_MODE_MASK	
Bit 3=1	for block search via program test (SERUPRO)
Bit 4=1	for each REPOS

Note

If neither bit 3 nor bit 4 is set, "non-path axes" are repositioned in the residual block in this phase.

Prefer or ignore REPOS

Further REPOS adaptations can be made by setting the bits in:

MD11470 $MN_REPOS_MODE_MASK

Bit 5 = 1 Modified feedrates and spindle speeds are valid immediately in the residual block and are given priority. This behavior relates to every REPOS operation.

Bit 6 = 1 Neutral axes and positioning spindles are repositioned after SERUPRO.

Neutral axes that are not allowed to be further repositioned must receive interface signal:
DB31, ... DBX10.0 (REPOSDELAY)

This cancels the REPOS motion.

Bit 7 = 1 The level of interface signal:
DB31, ... DBX10.0 (REPOSDELAY)
is read if REPOSA is interpreted.

Axes, which are neither geometry nor orientation axes, are then excluded by REPOS and are not moved.

Remark: REPOSDELAY is changed from edge to level evaluation.

Delayed approach of axis with REPOS offset

With the axial level-triggered NC/PLC interface signal axis/spindle (PLC→NCK):
DB31, ... DBX10.0 (REPOSDELAY)

the REPOS offset for this axis is traversed only
after the next time it is programmed with the edge of IS:
DB21, ... DBX31.4 (REPOSMODEEDGE).

Whether this axis is currently subject to a REPOS offset can be scanned via synchronized actions with $AA_REPOS_DELAY.

⚠ CAUTION

Interface signal:
DB31, ... DBX10.0 (REPOSDELAY)
has no effect on machine axes that form a path.

Whether an axis is a path axis can be determined with:
DB31, ... DBX76.4 (path axis).

Acceptance timing of REPOS NC/PLC interface signals

With the 0/1 edge of the channel-specific NC/PLC interface signal (PLC→NCK):
DB21, ... DBX31.4 (REPOSMODEEDGE)

the level signals of:
DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2)

and
DB31, ... DBX10.0 (REPOSDELAY)

are transferred to the NC.

The levels relate to the current block in the main run. There are two different cases:

Case A:

One repositioning block of a currently active REPOS operation is contained in the main run.

The active REPOS operation is aborted, restarted and the REPOS offsets controlled via the signals:
DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2)

and
DB31, ... DBX10.0 (REPOSDELAY).

Case B:

No repositioning block of a currently active REPOS operation is contained in the main run.

Each future REPOS operation wishing to reapproach the current main program block is controlled by the level of interface signal:
DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2)

and
DB31, ... DBX10.0 (REPOSDELAY).

Note

In the running ASUB, the IS:
DB21, ... DBX31.4 (REPOSMODEEDGE)
does not affect the final REPOS, unless this signal applies to the REPOS blocks.

In Case A, the signal is only allowed in the stopped state.

Response to RESET:

NCK has acknowledged the PLC signal

If the level of the signals:
DB21, ... DBX31.4 (REPOSMODEEDGE) = 1
and
DB21, ... DBX319.0 (REPOSMODEEDGEACKN) = 1

and
a RESET occurs in this situation, then the interface signal of the NCK:
DB21, ... DBX319.1–319.3 (Repos Path Mode Ackn0 to 2)
is deleted.

NCK has not yet acknowledged the PLC signal:

If the level of the signals:
DB21, ... DBX31. (REPOSMODEEDGE 4) = 1
and
DB21, ... DBX319.0 (REPOSMODEEDGEACKN) = 0
and if a RESET occurs in this situation, the NCK deletes
DB21, ... DBX319.0 (REPOSMODEEDGEACKN) = 0
and
DB21, ... DBX319.1–319.3 (Repos Path Mode Ackn0 to 2)
.

Controlling SERUPRO approach with NC/PLC interface signals

The SERUPRO approach can be used with:
DB21, ... DBX31.4 (REPOSMODEEDGE)
and the associated signals in the following phases:

- Between "Search target found" and "Start SERUPRO ASUB"
- From "SERUPO-ASUB stops automatically before REPOS" to "Target block is executed"

While the SERUPRO ASUB is being executed, e.g. in the program section before the REPOS operation, interface signal:
DB21, ... DBX31.4 (REPOSMODEEDGE)
does not affect the SERUPRO approach.

REPOS operations with NC/PLC interface signals

Control REPOS with NC/PLC interface signals

REPOS offsets can be positively influenced with the following channel-specific NC/PLC interface signals **from the PLC**:

DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2) channel-specific *

DB21, ... DBX31.4 (REPOSMODEEDGE) channel-specific

DB31, ... DBX10.0 (REPOSDELAY) *axis/spindle
(This axial interface does **not** affect machine axes that form a path.)

DB31, ... DBX72.0 (REPOSDELAY) axis/spindle

* These signals are available in the respective DB of the HMI or PLC.

REPOS acknowledgement signals

The following NC/PLC interface signals can be used to acknowledge **from the NCK**, functions that control the REPOS response via PLC:

DB21, ... DBX319.0 (REPOSMODEEDGEACKN) channel-specific

DB21, ... DBX319.1-319.3 (Repos Path Mode Ackn0 to 2) channel-specific

DB21, ... DBX319.5 (Repos DEFERRAL Chan) channel-specific

DB31, ... DBX70.0 (Repos offset) axis/spindle

DB31, ... DBX70.1 (Repos offset valid) axis/spindle

DB31, ... DBX70.2 (Repos Delay Ackn) axis/spindle

DB31, ... DBX76.4 (path axis) axis/spindle

For further information, see "REPOS offset in the interface"

REPOS acknowledgement operations

With the channel-specific NC/PLC interface signal:
DB21, ... DBX319.0 (REPOSMODEEDGEACKN)

a "handshake" is established by the interface signal:
DB21, ... DBX31.4 (REPOSMODEEDGE)

which is recognized by the NC and acknowledged with DB21, ... DBX319.0 to the PLC.

Note

If the NCK has not yet acknowledged interface signal:
DB21, ... DBX31.4 (REPOSMODEEDGE)
with interface signal:
DB21, ... DBX319.0 (REPOSMODEEDGEACKN)
a RESET in this situation causes the program to abort, and the REPOS that is to be used to control the REPOSPATHMODE can no longer take place.

A REPOSMODE specified by the PLC is acknowledged by the NCK with the interface signals:
DB21, ... DBX319.1-319.3 (Repos Path Mode Ackn0 to 2)

and
DB31, ... DBX10.0 (Repos Delay)

with:
DB31, ... DBX70.2 (Repos Delay Ackn)

in the following way:

A part program is stopped at `N20` (→ time (2) in figure). The NCK stops according to the braking ramp. After the PLC has specified the REPOSPATHMODE, the NCK accepts the REPOSPATHMODE with the 0/1 edge of REPOSMODEEDGE at → Time (3). Repos Path Mode Ackn remains set until the ASUB is initiated (→ Time (4)). The REPOS command is started in the ASUB (→ Time (5)). The ASUB RESET block is activated again (→ Time (6)):

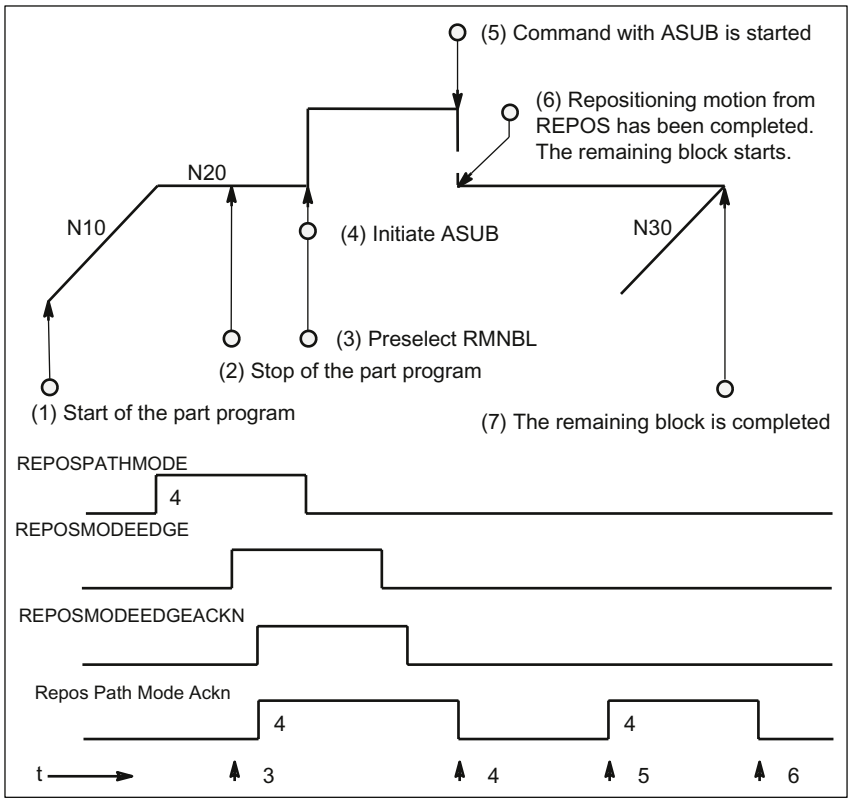


Figure 8-6 REPOS sequence in part program with timed acknowledgement signals from NCK

NCK sets acknowledgement again

Phase with REPOSPATHMODE still active (residual block of the program stopped at → Time (2) is not yet completely executed).

As soon as the REPOS repositioning motion of the ASUB is executed, the NCK sets the "Repos Path Mode Ackn" again (→ Time (5)). If no REPOSPATHMODE has been preselected via an NC/PLC interface signal, the programmed REPOS mode is displayed.

"Repos Path Mode Ackn" is canceled when the residual block is activated (→ Time (6)). The part program block `N30` following the block at → Time (2) is resumed.

Interface signal:
DB31, ... DBX70.2 (Repos Delay Ackn) is defined in the same way.

DB31, ... DBX70.1 (Repos offset valid) = 1, if:

DB21, ... DBX319.1-319.3 (Repos Path Mode Ackn0 to 2) = 4 (RMNBL).

Valid REPOS offset

At the end of the SERUPRO operation, the user can read out the REPOS offset via the axis/spindle NC/PLC interface signal (NCK→PLC):
DB31, ... DBX70.0 (REPOS offset).

The effects of this signal on the relevant axis are as follows:

Value 0: No REPOS offset is applied.
Value 1: REPOS offset is applied.

Range of validity

Interface signal:
DB31, ... DBX70.0 (REPOS offset)
is supplied at the end of the SERUPRO operation.

The REPOS offset is invalidated at the start of a SERUPRO ASUB or the automatic ASUB start.

Updating the REPOS offset within the scope

Between the SERUPRO end and SERUPRO start, the axis can be moved in JOG mode with a mode change.

In JOG mode, the user manually moves the axis over the REPOS offset path in order to set interface signal:
DB31, ... DBX70.0 (REPOS offset) to the value 0.

Within the range of validity, the axis can also be traversed using FC18, whereby the IS DB31, ... DBX70.0 (REPOS offset) is continuously updated.

Displaying the range of validity

The range of validity of the REPOS offset is indicated with interface signal:

DB31, ... DBX70.1 (REPOS offset valid)

It is indicated whether the REPOS offset calculation was valid or invalid:

Value 0: The REPOS offset of this axis is calculated correctly.
Value 1: The REPOS offset of this axis cannot be calculated, as the REPOS has not yet occurred, e.g. it is at the end of the ASUB, or no REPOS is active.

REPOS offset after an axis interchange

The group signal:
DB21, ... DBX319.5 (Repos DEFERRAL Chan)
can be used to determine whether a valid REPOS offset has taken place:

Value 0: All axes currently controlled by this channel have either no REPOS offset or their REPOS offsets are invalid.

Value 1: Miscellaneous.

REPOS offset with synchronized synchronous spindle coupling

When repositioning with SERUPRO, processing continues at the point of interruption. If a synchronous spindle coupling was already synchronized, there is no REPOS offset of the following spindle and no synchronization path is present. The synchronization signals remain set.

Search target found on block change

The axial NC/PLC interface signal:
DB31, ... DBX76.4 (path axis)
is 1 if the axis is part of the path grouping.

This signal shows the status of the current block to be executed during block change. Subsequent status changes are ignored.

If the SERUPRO operation is ended with "Search target found",
DB31, ... DBX76.4 (path axis) refers to the target block.

8.8.2.2 Repositioning on contour with controlled REPOS

Approach modes

Influence path axes individually

During SERUPRO approach, a REPOS operation is initiated in order to reposition to the contour. A large number of axes which the user can control by means of interface signals is frequently moved. The operator panel interface supplies the offsets per channel axis which REPOS intends to traverse.

Repositioning of the individual path axes can be controlled by the PLC with the interface signals and it therefore has priority over the actual commands `RMIBL`, `RMBBL` and `RMEBL` in the part program.

`RMIBL` Repositioning to interruption point

`RMBBL` Repositioning to start-of-block

`RMEBL` Repositioning to end-of-block position

`RMNBL` Repositioning to next point on path

Repositioning with RMNBL

Like RMIBL, RMBBL and RMEBL, RMNBL (REPOS Mode Next) **is redefined** for SERUPRO approach. After an interruption, RMNBL does not to complete an already started repositioning, but processes from the next path point:

At the time REPOSA is interpreted, position (B) is referenced in order to find point C at the interruption block with the shortest distance to B. The repositioning block moves from B to C to the end position.

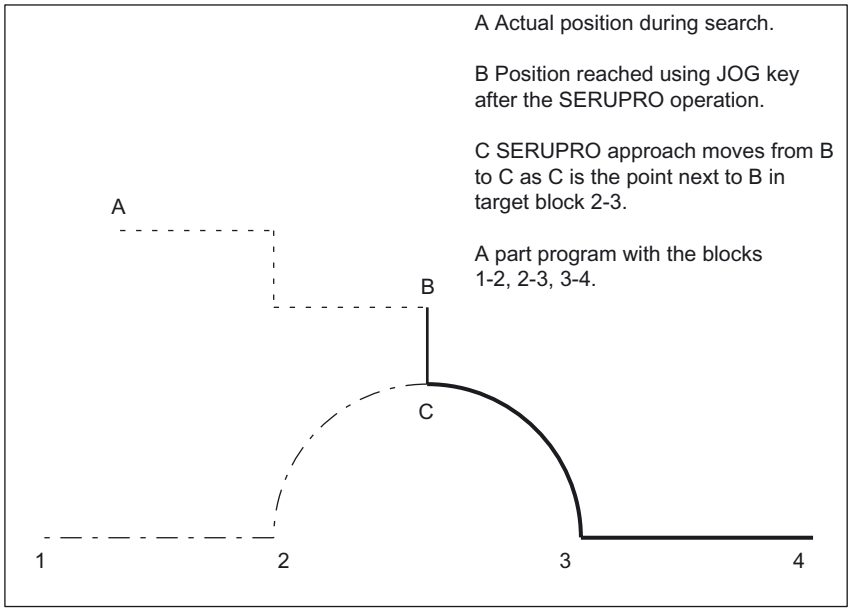


Figure 8-7 SERUPRO approach under RMNBL

Application and procedure

SERUPRO approach with RMNBL offers the opportunity of using the application shown in the figure: If a program abort is forced by RESET at any point when the program is advancing from block 2 to 3,

- RMNBL is used to approach the abort location by the shortest route in order to process just the distance-to-go from C-3 and 3-4. The user starts a SERUPRO operation at the interruption block and uses the JOG keys to move in front of the faulty position of the target block.
- RMIBL and RMBBL are always approached with B-2, 2-3, 3-4 and the target block thus repeated once completely.

Selecting REPOS mode

Using the three bits of the channel-specific NC/PLC interface signal (PLC→NCK) DB21, ... DBX31.0-31.2 (REPOSPATHMODE0-2), the respective function RMBBL, RMIBL, RMEBL or RMNBL can be selected.

Repositioning point

RMNOTDEF REPOS mode is not redefined
RMBBL Repositioning to block start point or last end point
RMIBL Repositioning to interruption point
RMEBL Repositioning to end-of-block point
RMNBL Repositioning to next path point

DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=0 corresponds to **RMNOTDEF**

DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=1 corresponds to **RMBBL**

DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=2 corresponds to **RMIBL**

DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=3 corresponds to **RMEBL**

DB21, ... DBX31.0-31.2 (REPOSPATHMODE)=4 corresponds to **RMNBL**

With DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2) = 0, no settings are overwritten and the current program is valid. The interface signal responds to the level of the corresponding mode.

Note

RMNBL is a general REPOS extension and it is not restricted to SERUPRO. For SERUPRO RMIBL and RMBBL are identical.

With DB21, ... DBX31.0–31.2 (REPOSPATHMODE0 to 2), the path as a whole is controlled. The path axes cannot be changed individually.

The behavior of the other axis types can be changed individually using interface signal DB31, ... DBX10.0 (REPOSDELAY). This REPOS offset is not applied immediately, but only when it is next programmed.

For further information on the programming of the repositioning point, see:

References:
Programming Manual, Job Planning; Path Behavior, Section: Repositioning on contour

Read REPOS mode in synchronized actions

The valid REPOS mode of the interrupted block can be read via synchronized actions using system variable $AC_REPOS_PATH_MODE=

0: Not defined	Repositioning not defined
1: RMBBL	Repositioning to beginning
2: RMIBL	Repositioning to point of interruption
3: RMEBL	Repositioning to block end point
4: RMNBL	Repositioning to next geometric path point of interrupted block

8.8.3 Acceleration measures via MD

Machine data settings

The processing speed of the entire SERUPRO operation can be accelerated using the following machine data.

MD22600 $MC_SERUPRO_SPEED_MODE and

MD22601 $MC_SERUPRO_SPEED_FACTOR

With MD22600 $MC_SERUPRO_SPEED_MODE == 1, the SERUPRO operation will run at the usual "dry run feedrate".

Through MD22600 $MC_SERUPRO_SPEED_MODE == 0

MD22601 $MC_SERUPRO_SPEED_FACTOR is evaluated,

and a further acceleration is permitted. Dynamic monitoring functions are disabled in this mode.

SPEED factor for channel axes during ramp-up

Machine data MD22600 $MC_SERUPRO_SPEED_MODE is effective for the following channel axes in the main run throughout the entire SERUPRO operation:

- PLC axes
- Command axes
- Positioning axes
- Reciprocating axes

The functions of MD22600 $MC_SERUPRO_SPEED_MODE and MD22601 $MC_SERUPRO_SPEED_FACTOR apply only to SERUPRO and not to program testing. In this case **no axes/spindles are moved**.

CAUTION
The NC as a discrete system generates a sequence of interpolation points. It is possible that a synchronized action that was triggered in normal operation will no longer be triggered in SERUPRO.

Mode of functioning with DryRun

An active SERUPRO SPEEDFACTOR has the following effect on DryRun:

- DryRun is activated simultaneously.

  This causes a switch from `G95/G96/G961/G97/G971` to `G94` in order to execute `G95/G96/G961/G97/G971` as quickly as you wish.

- Tapping and thread cutting are performed at the usual velocity for DryRun.

DryRun and SERUPRO affect the spindle/axis with the following G codes:

- G331/G332 causes the spindle to be interpolated as an axis in a path grouping. In the case of tapping, the drilling depth (e.g. axis X) and the pitch and speed (e.g. spindle S) are specified.

  In the case of DryRun, the velocity of X is specified, the speed remains constant, and the pitch is adjusted.

  Following the SERUPRO simulation, the position for spindle S deviates from the normal position because the spindle S has rotated less during simulation.

8.8.4 SERUPRO ASUB

SERUPRO ASUB special points

Special points should be noted for SERUPRO ASUB with regard to:

- Reference point approach: Referencing via part program `G74`
- Tool management: Tool change and magazine data
- Spindle ramp-up: On starting a SERUPRO ASUB

G74 reference point approach

If statement `G74` (reference point approach) is programmed between the program start and the search target, this will be ignored by the NC.

SERUPRO approach does not take this `G74` statement into account!

Tool management

If tool management is active, the following setting is recommended:

Set MD18080 $MA_TOOL_MANAGEMENT_MASK BIT 20 = 0.

The tool management command generated during the SERUPRO operation is thus **not** output to the PLC!
The tool management command has the following effect:

- The NC acknowledges the commands automatically.
- No magazine data is changed.
- Tool data is not changed.
  Exception:
  The tool enabled during the test mode can assume 'active' state. In this way, the wrong tool may be on the spindle after the SERUPRO operation.
  Remedy:
  The user starts a SERUPRO ASUB that is actually traversed. Prior to the start, the user can start an ASUB that loads the correct tool.

SERUPRO operation: Functionality: In sequence steps 2. to 6.
SERUPRO ASUB: Functionality: The sequence of step 7.

In addition, machine data setting
MD18080 $MA_TOOL_MANAGEMENT_MASK Bit 11 = 1 is required because the ASUB may have to repeat a T selection.

Systems with tool management and auxiliary spindle are not supported by SERUPRO!

Example

Tool change subprogram

```
Program code                                    Comment
PROC L6                                         ; Tool change routine
N500 DEF INT TNR_AKTUELL                        ; Variable for active T number
N510 DEF INT TNR_VORWAHL                        ; Variable for preselected T number

                                                ; Determine current tool
N520 STOPRE                                     ; In program testing
N530 IF $P_ISTEST                               ; from the program context
N540 TNR_AKTUELL = $P_TOOLNO                    ; the "current" tool is read.
N550 ELSE                                       ; Otherwise, the tool of the spindle is
                                                  read out.
N560 TNR_AKTUELL = $TC_MPP6[9998,1]             ; Read tool T number on the spindle
N570 ENDIF

N580 GETSELT(TNR_VORWAHL)                       ; Read T number of the preselected tool
                                                  of the master spindle. Execute tool
                                                  change only if tool is not yet
                                                  current.
N590 IF TNR_AKTUELL <> TNR_VORWAHL              ; Approach tool change position
N600 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0
N610 M206                                       ; Execute tool change
N620 ENDIF
N630 M17
```

ASUB for calling the tool change routine after block search type 5

```
Program code                                    Comment
PROC ASUPWZV2
N1000 DEF INT TNR_SPINDEL                       ; Variable for active T number
N1010 DEF INT TNR_VORWAHL                       ; Variable for preselected T number
N1020 DEF INT TNR_SUCHLAUF                      ; Variable for T number determined in
                                                  block search
N1030 TNR_SPINDEL = $TC_MPP6[9998,1]            ; Read tool T number on the spindle
N1040 TNR_SUCHLAUF = $P_TOOLNO                  ; Read T number determined by search
                                                  run, i.e. that tool determines the
                                                  current tool offset.
N1050 GETSELT(TNR_VORWAHL)                      ; Read T number of preselected tool
N1060 IF TNR_SPINDEL ==TNR_SUCHLAUF GOTOF       ;
ASUP_ENDE1
N1070 T = $TC_TP2[TNR_SUCHLAUF]                 ; T selection by tool name
N1080 L6                                        ; Call tool change routine
                                                ;
N1085 ASUP_ENDE1:                               ;
N1090 IF TNR_VORWAHL == TNR_SUCHLAUF GOTOF      ;
ASUP_ENDE
N1100 T = $TC_TP2[TNR_VORWAHL]                  ; Restore T preselection by tool name
                                                ;
N1110 ASUP_ENDE:                                ;
N1110 M90                                       ; Feedback to PLC
N1120 REPOSA                                    ; ;ASUB end
```

In both of the programs PROC L6 and PROC ASUPWZV2, the tool change is programmed with M206 instead of M6.
ASUB program "ASUPWZV2" uses different system variables to detect the progress of the program ($P_TOOLNO) and represent the current status of the machine ($TC_MPP6[9998,1]).

Spindle ramp-up

When the SERUPRO ASUB is started, the spindle is not accelerated to the speed specified in the program because the SERUPRO ASUB is intended to move the new tool into the correct position at the workpiece after the tool change.

A spindle ramp-up is performed with SERUPRO ASUB as follows:

- SERUPRO operation has finished completely.
- The user starts the SERUPRO ASUB via function block FC 9 in order to ramp up the spindle.
- The start after M0 in the ASUB does not change the spindle status.
- SERUPRO ASUB automatically stops before the REPOS part program block.
- The user presses START.
- The spindle accelerates to the target block state if the spindle was not programmed differently in the ASUB.

Note

Modifications for REPOS of spindles:

The transitions of speed control mode and positioning mode must be taken into consideration in the event of modifications in SERUPRO approach and spindle functionality.

For further information on the operating mode switchover of spindles, see Section "Modes (Page 1384)".

8.8.5 Selfacting SERUPRO

Selfacting SERUPRO

The channel-specific function "Self-acting SERUPRO" allows a SERUPRO sequence **without** having to previously define a search target in a program of the associated SERUPRO channels.

In addition, a special channel, the "serurpoMasterChan", can be defined for **each** "Self-acting SERUPRO". A search target can be defined in this channel.

The "Selfacting SERUPRO" function supports the SERUPRO cross-channel block search.

Function

The "SelfActing SERUPRO" operation cannot be used to find a search target. If the search target is not reached, no channel is stopped. In certain situations, however, the channel is nevertheless stopped temporarily. In this case, the channel will wait for another channel. Examples are: Wait marks, couplings, or axis replacement.

Wait phase occurs:

During this wait phase, the NC checks whether the channel "seruproMasterChan" has reached a search target. If no search target is reached, the Wait phase is left.

If the search target is reached,
the SERUPRO operation is also ended in the channel. The "seruproMasterChan" channel must have been started in normal SERUPRO mode.

No wait phase occurs:

"Self-Acting SERUPRO" is ended by M30 of the part program.
The channel is now in Reset state again.
A SERUPRO approach does not not take place.

Starting a group of channels

If a group of channels is only started with "SelfActing SERUPRO", then all channels are ended with "RESET".

Exceptions:
A channel waits for a partner channel that has not been started at all.

A cross-channel block search can be carried out as follows:

- Via the HMI, the user selects the channels that must work together (channel group).
- The user chooses an especially important channel from the channel group for which he wants to select a search target explicitly (target channel).
- The HMI will then start SERUPRO on the target channel and "SelfActing SERUPRO" in the remaining channels of the channel group.

The operation is complete if **each** channel concerned has deleted "seruproActive".

"Selfacting SERUPRO" accepts no master channel on another NCU.

Activation

"Self-acting SERUPRO" is activated via the HMI as a block search start for the Type 5 block search for target channel "seruproMasterChan".

No search target is specified for dependent channels started from the target channel.

8.8.6 Locking a program section for "Continue machining at the contour"

Programmed interrupt pointer

If because of manufacturing and/or process-related reasons, "Continue machining at the contour" may not be possible within a certain program section at a program abort, this program section can be locked for the target block of a block search.

If after a program abort there is a block search for the interruption point within the locked program section for "Continue machining at the contour", the last executable block (main run block) before the start of the locked section is used by the control as target block (hold block).

Programming

Syntax

```
IPTRLOCK()
```

Functionality

Marks the beginning of the program section as of which "Continue machining at the contour" is locked. The next executable block (main run block) in which IPTRLOCK becomes active is now used as target block for a block search with "Continue machining at the interruption point", until the release with IPTRUNLOCK. This block is referred to as the **hold block** in the following.

Effectiveness: Modal

Syntax

```
IPTRUNLOCK()
```

Marks the end of the program section locked for "Continue machining at the contour". As of the next executable block (main run block) in which IPTRLOCK becomes active, the current block is used again as target block for a block search with "Continue machining at the interruption point". This block is referred to as the **release block** in the following.

Effectiveness: Modal

Example

```
Program code                    Comment
...
N010 IPTRLOCK()                 ; Locked area: Start
N020 R1=R1+1
N030 G4 F1                      ; Hold block
...                             ; Locked area
N200 IPTRUNLOCK()               ; Locked area: End
N220 R1=R1+1
N230 G4 F1                      ; Release block
...
```

Supplementary conditions

- IPTRLOCK acts within a program (*.MPF, *.SPF) at the most up to the end of the program (M30, M17, RET). IPTRUNLOCK implicitly becomes active at the end of the program.
- Multiple programming of IPTRLOCK within a program does not have a cumulative effect. With the first programming of IPTRUNLOCK within the program or when the end of the program is reached, all previous IPTRLOCK calls are terminated.
- If there is a subprogram call within a locked area, "Continue machining at the contour" is also locked for this and all following subprogram levels. The lock also cannot be cancelled within the called subprogram through explicit programming of IPTRUNLOCK.

Example: Nesting of locked program sections in two program levels

With the activation of the "Continue machining at the contour" lock in PROG_1, "Continue machining at the contour" is also locked for PROG_2 and all following program levels.

Program code	Comment
PROC PROG_1	; Program 1
...	
N010 IPTRLOCK()	
N020 R1=R1+1	
N030 G4 F1	**; Hold block**
...	; Locked area: Start
N040 PROG_2	; Locked area
...	; Locked area: End
N050 IPTRUNLOCK()	
N060 R2=R2+2	
N070 G4 F1	**; Release block**
...	

Program code	Comment
PROC PROG_2	; Program 2
N210 IPTRLOCK()	; Ineffective due to program 1
...	
N250 IPTRUNLOCK()	; Ineffective due to program 1
...	
N280 RET	; Ineffective due to program 1

Example 3: Multiple programming of IPTRLOCK

```
Program code                    Comment
PROC PROG_1                     ; Program 1
...
N010 IPTRLOCK()
N020 R1=R1+1
N030 G4 F1                      ; Hold block
...                             ; Locked area: Start
N150 IPTRLOCK()                 ; Locked area
...                             ; Locked area
N250 IPTRLOCK()                 ; Locked area
...                             ; Locked area: End
N360 IPTRUNLOCK()
N370 R2=R2+2
N380 G4 F1                      ; Release block
...
```

System variable

The status of the current block can be determined via the system variable $P_IPTRLOCK:

$P_IPTRLOCK	Meaning
FALSE	The current block is **not** within a program section locked for "Continue machining at the contour"
TRUE	The current block is within a program section locked for "Continue machining at the contour"

Automatic function-specific "Continue machining at the contour" lock

For various couplings, the activation/deactivation of the "Continue machining at the contour" lock can be performed channel-specifically automatically with the activation/deactivation of the coupling:

MD22680 $MC_AUTO_IPTR_LOCK, bit x

Bit	Value	Meaning
0	Electronic gear (EGON / EGOF)	
	1	Automatic "Continue machining at the contour" lock is active
	0	Automatic "Continue machining at the contour" lock is not active
1	Axial master value coupling (LEADON / LEADOF)	
	1	Automatic "Continue machining at the contour" lock is active
	0	Automatic "Continue machining at the contour" lock is not active

This program section begins with the last executable block **before** the activation and ends with the deactivation.

The automatic interrupt pointer is not active for couplings that were activated or deactivated via synchronized actions.

Example: Automatically declaring axial master value coupling as search-suppressed:

```
Program code                          Comment
N100 G0 X100
N200 EGON(Y,"NOC",X,1,1)              ; Search-suppressed program section starts.
N300 LEADON(A,B,1)
...
N400 EGOFS(Y)
...
N500 LEADOF(A,B)                      ; Search-suppressed program section ends.
N600 G0 X200
```

A program abort within search-suppressed program section (N200 - N500) always provides the interrupt pointer with N100.

CAUTION

If there is an overlap of the "Programmable interrupt pointer" and "Automatic interrupt pointer" functions via machine data, the NC selects the largest possible search-suppressed area.

A program may need a coupling for almost all of the runtime. In this case, the automatic interrupt pointer would always point to the start of the program and the SERUPRO function would in fact be useless.

8.8.7 Special features in the part program target block

8.8.7.1 STOPRE in the target block of the part program

STOPRE block

The STOPRE block receives all modal settings from the preceding block and can, therefore, apply conditions in advance in relation to the following actions:

- Synchronize program line currently processing with the main run.
- Derive modal settings for SERUPRO in order, for example, to influence this REPOS motion on approach of SERUPRO.

Example 1:

Position a Z axis by specifying an X axis setpoint.

When block "G1 F100 Z=$AA_IM[X]" is interpreted, the preceding STOPRE block ensures synchronization with the main run. The correct setpoint of the X axis is thus read via $AA_IM to move the Z axis to the same position.

Example 2:

Read and correctly calculate external zero offset.

Program code	Comment
N10 G1 X1000 F100	;
N20 G1 X1000 F500	;
N30 G1 X1000 F1000	;
N40 G1 X1000 F5000	;
N50 SUPA G1 F100 X200	; move external zero offset after 200
N60 G0 X1000	;
N70 ...	;

Via an implicit STOPRE before N50, the NCK can read and correctly calculate the current zero offset.
For a SERUPRO operation on the N50 search target, repositioning is on the implicit STOPRE in the SERUPRO approach and the velocity is determined from N40 with F5000.

Implicit preprocessing stop

Situations in which Interpreter issues an implicit preprocessing stop:

1. In all blocks in which one of the following variable access operations occurs: -
   Programming of a system variable beginning with $A...
   -Redefined variable with attribute SYNR/SYNRW
2. For the following part program commands:
   -Part program command MEACALC, MEASURE
   -Programming of SUPA (suppress frames and online offsets)
   -Programming of CTABDEF (start of curve table definition)
   -Part program command WRITE/DELETE (write/delete file)
   -Before the first WRITE/DELETE command in a sequence of such commands
   -Part program command EXTCALL
   -Part program command GETSELT, GETEXET
   -Tool change and active fine tool offset FTOCON
3. For the following command execution:
   -Finishing of Type 1 search ("Search without calculation") and
   Type 2 search with calculation ("Search at contour end point")

   Note: Type 2 search "Block search at contour start point" has the same behavior.

8.8.7.2 SPOS in target block

SPOS

If a spindle is programmed with M3/M4 and the target block contains an SPOS command, the spindle is switched over to SPOS on completion of the SERUPRO process (search target located). This is indicated on the VDI interface.

8.8.8 Behavior during POWER ON, mode change and RESET

SERUPRO is inactive during POWER ON. The mode change is permitted during SERUPRO. RESET will cancel SERUPRO and deselects the internally selected program test. SERUPRO cannot be combined with other block search types.

8.8.9 Supplementary conditions

8.8.9.1 Travel to fixed stop (FXS)

During repositioning (REPOS), the "Travel to fixed stop" function (FXS) is repeated automatically. Every axis is taken into account. The torque programmed last before the search target is used as torque.

System variable

The system variables for "Travel to fixed stop" have the following meaning with SERUPRO:

- $AA_FXS: Progress of the program simulation
- $VA_FXS: Real machine state

The system variables always have the same values **outside** of the SERUPRO function.

ASUB

A user-specific ASUB can be activated for SERUPRO.

References

For detailed information on the SERUPRO block search, see Section "General functionality (Page 304)".

8.8.9.2 Travel with limited torque/force (FOC)

During repositioning (REPOS), the "Travel with limited torque/force" function (FOC) is repeated automatically. Every axis is taken into account. The torque programmed last before the search target is used as torque.

System variable

The system variables for "Travel with limited torque/force" have the following meaning with SERUPRO:

- $AA_FOC: Progress of the program simulation
- $VA_FOC: Real machine state

Supplementary condition

A changing torque characteristic **cannot** be implemented during repositioning.

Example

A program traverses axis X from 0 to 100 and switches "Travel with limited torque/force" (FOC) on every 20 increments for 10 increments. This torque characteristic is usually generated with non-modal FOC and cannot be performed during repositioning (REPOS). Instead, axis X is traversed from 0 to 100 with or without limited torque/force in accordance with the last programming.

References

For detailed information on the SERUPRO block search, see Section "General functionality (Page 304)".

8.8.9.3 Synchronous spindle

The synchronous spindle can be simulated.

The synchronous spindle operation with main spindle and any number of following spindles can be simulated in all existing channels with SERUPRO.

For further information about synchronous spindles, see:

References:

Function Manual, Extended Functions; Synchronous Spindle (S3)

8.8.9.4 Couplings and master-slave

Setpoint and actual value couplings

The SERUPRO operation is a program simulation in Program Test mode with which setpoint and actual value couplings can be simulated.

Specifications for EG simulation

For simulation of EG, the following definitions apply:

1. Simulation always takes place with setpoint coupling.
2. If not all leading axes are under SERUPRO, the simulation is aborted with Alarm 16952 "Reset Clear/No Start". This can occur with cross-channel couplings.
3. Axes that have only one encoder from the NCK point of view and are moved externally, cannot be simulated correctly. These axes must not be integrated in couplings.

⚠ CAUTION

In order to be able to simulate couplings correctly, they must have been

switched off previously.

This can be performed with machine data MD10708 $MA_SERUPRO_MASK.

Specifications for coupled axes

The SERUPRO operation simulates coupled axes always assuming that they are setpoint couplings. In this way, the end points are calculated for **all** axes that are used as target points for SERUPRO approach. The coupling is also active with "Search target found". The path from the current point to the end point is carried out for SERUPRO approach with the active coupling.

LEADON

The following specifications apply for the simulation of axial master value couplings:

1. Simulation always takes place with setpoint coupling.
2. SERUPRO approach takes place with active coupling and an overlaid motion of the following axis in order to reach the simulated target point.

The following axis that is moved solely by the coupling cannot always reach the target point. In SERUPRO approach, an overlaid linear motion is calculated for the following axis to approach the simulated point!

Reaching simulated target point for LEAD with JOG

At the time of "Search target found", the coupling is already active, especially for the JOG motions. If the target point is not reached, SERUPRO approach can be used to traverse the following axis with active coupling and an overlaid motion to the target point.

Note

For further information on the repositioning of axis couplings, see Section "Continue machining at the contour after SERUPRO search target found (Page 541)".

Master-slave

A system ASUB can be started automatically after the block search is finished. In this subprogram, the user can control the coupling state and the associated axis positions subsequently. The required information is provided via the following system variables:

System variable	Description
$P_SEARCH_MASLD[<slave axis>]	Position offset between slave and master axis when the link is closed.
$AA_MASL_STAT[<slave axis>]	Current state of a master/slave coupling
$P_SEARCH_MASLC[<slave axis>]	Status: The state of the coupling was changed during the block search
The system variables are deleted when the coupling is switched on (MASLON).	

Note

The coupled axes must be in the same channel when the block search is executed.

Further information on the master/slave coupling can be found in:

References:
Function Manual, Special Function; Speed/Torque Coupling (TE3) Master-Slave

The system ASUB is called progevent.spf and must be available in the /_N_CMA_DIR directory. The contents might be as follows:

progevent.spf
X=Master axis, Y=Slave axis

Programming

```
N10 IF(($S_SEARCH_MASLC[Y]< >0) AND ($AA_MASL_STAT[Y]< >0))
N20 MASLOF(Y)
N30 SUPA Y=$AA_IM[X]-$P_SEARCH_MASLD[Y]
N40 MASLON(Y)
N50 ENDIF
N60 REPOSA
```

To ensure that the ASUB can be automatically started, the following machine data must be set:

- NC-specifically:
  - MD11602 $MN_ASUP_START_MASK = 'H01'
  - MD11604 $MN_ASUP_START_PRIO_LEVEL = 100
  - MD11450 $MN_SEARCH_RUN_MODE = 'H02'
- Channel-specifically for the channel in which the ASUB is started or generally for all channels:
  - MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK = ’H3F’
  - MD20115 $MC_IGNORE_REFP_LOCK_ASUP = ’HFFFFFFFF’

Axis couplings

Note

For a leading axis whose following axes are in another channel, the setting for acceleration of the processing speed has no effect:
MD22601 $MC_SERUPRO_SPEED_FACTOR > 0

Coupled motion

The coupled motion function (TRAILON) is supported by SERUPRO.

For further information on coupled motion with TRAILON and TRAILOF, see:

References:

- Function Manual, Special Functions; Axis Couplings (M3)
- Programming Manual, Job Planning; Axis Couplings

Gantry axes

The gantry axis function is supported by SERUPRO.

For further information on the functionality of gantry axes, see:

References:
Function Manual, Special Functions; Gantry Axes (G1)

Tangential control

The tangential follow-up of individual axes function is supported by SERUPRO.

For further information on tangential control, please see:

References:
Function Manual, Special Functions; Tangential Control (T3)

8.8.9.5 Axis functions

SERUPRO conditions

The special conditions for SERUPRO must be observed with axis enable, autonomous axis operations, and axis replacement.

Axis enable

The axial interface DB31, ... DBX3.7 ("Program test axis/spindle enable") controls the axis enables if no closed-loop controller enable is to (or can) be issued at the machine and is active only during the program test or when SERUPRO is active.

It is possible to issue this enable via interface signal PLC→NCK
DB31, ... DBX3.7 (program test axis/spindle enable). If the real servo enable is missing during program testing or SERUPRO, the effect on the axes/spindles is as follows:

- As soon as the simulated program run intends to move an axis/spindle, the message "Waiting for axis enable" or "Waiting for spindle enable" is displayed and the simulation is stopped.
- If during a simulated motion, NC/PLC interface signal DB31, ... DBX3.7 (program test axis/spindle enable) is then canceled, alarm 21612: "Channel %1 axis %2 NC/PLC interface signal ’controller enable’ reset during motion” is activated.

Autonomous axis operations

Autonomous single-axis operations are axes controlled by the PLC that can also be simulated on SERUPRO. During SERUPRO operation, as in normal operation, the PLC can take over or give up control of an axis. If required, this axis can also be traversed using FC18. The PLC takes over control of the axis **before** the approach block and is responsible for positioning this axis. This is valid for all block search types.

For further information about autonomous single-axis operations, see:

References:
Function Manual, Extended Functions; Positioning Axes (P2)

Axis replacement

Problem: A program moves an axis and gives up control before the target block with WAITP(X). X is thus not subject to REPOS and the axis is not taken into account in SERUPRO approach.

Via the machine data MD11470 $MN_REPOS_MODE_MASK, the following behavior can be achieved for SERUPRO-REPOS:

The neutral axes are moved as "command axes" in the SERUPRO-REPOS. The axis interpolates without a path context even it was last programmed as a path axis. In this scenario, the velocity results from MD32060 $MA_POS_AX_VELO. After SERUPRO approach, this axis is again neutral.

Neutral axes that are however not allowed to be repositioned must receive the axial NC/PLC interface signal “REPOSDELAY”. This deletes the REPOS movement.

Example:

After SERUPRO, one axis is deliberately moved in the synchronized action via technology cycles. The command axes are always moved in the approach block, never in the target block. The target block can only be changed if all command axes have been moved to the end.

⚠ CAUTION
The PLC-controlled axis is not repositioned! Axes enabled by RELEASE(X) before the target block are not repositioned.

8.8.9.6 Gear stage change

Operational sequences

The gear stage change (GSC) requires physical motions from the NCK in order to be able to engage a new gear.
In the SERUPRO operation, no gear stage change is required and is carried out as follows:

Some gears can only be changed when controlled by the NC, since either the axis must oscillate or a certain position must be approached beforehand.

The gear stage change can be suppressed selectively for DryRun, program test, and SERUPRO using bits 0 to 2 in MD35035 $MA_SPIND_FUNCTION_MASK.

The gear stage change must then be performed in REPOS; this will work even if the axis involved is to be in "speed control mode" at the target block. In other cases, the automatic gear stage change is denied with an alarm if, for example, the axis was involved in a transformation or coupling between the gear stage change and the target block.

Note

For further information on gear stage changes in DryRun, Program test and SERUPRO, see Section "S1: Spindles (Page 1383)".

8.8.9.7 Superimposed motion

Only SERUPRO

If "overlaid movements" are used, only the block search via program test (SERUPRO) can be used, since the overlaid movements are interpolated accordingly in the main run. This applies in particular to $AA_OFF.

Velocity profile instead of maximum axis velocity

During Program test, a velocity profile must be used, which allows "superimposed movements" to be interpolated during the main run. It is thus not possible to interpolate at the maximum axis velocity.

The axis velocity is set in "Dry run feedrate" mode using SD42100 $SC_DRY_RUN_FEED.

The velocity of the SERUPRO operation is selected using MD22600 $MC_SERUPRO_SPEED_MODE.

8.8.9.8 NC/PLC interface signals

REPOS offset available

If a REPOS offset has resulted for an axis during SERUPRO, this is displayed via the axial NC/PLC interface at the end of the SERUPRO operation:

DB31, ...DBX70.0 == 1 (REPOS offset available)

Validity of the REPOS offset

The REPOS offset becomes invalid at the start of a SERUPRO ASUB or NC start to resume the machining:

DB31, ... DBX70.1 == 1 (REPOS offset invalid)

The axis can be traversed manually in JOG mode or via the PLC user program using FC 18 between the end of the SERUPRO operation and NC start to resume the machining. If the REPOS offset is traversed completely, the interface signal is reset.

8.8.9.9 Making the initial settings more flexible

Initial setting/initial SERUPRO setting

Machine data MD20112 $MC_START_MODE_MASK defines the initial setting of the control for part program start with respect to the G codes (especially the current plane and settable zero offset), tool length compensation, transformation, and axis couplings. The special option exists for the SERUPRO operation of using
MD22620 $MC_ENABLE_START_MODE_MASK_PRT
to select an initial setting that differs from the normal part program start. The new setting must therefore been stored in:
MD22620 $MC_START_MODE_MASK_PRT

The meaning of the bits of MD22620 is identical to those of:
MD20112 $MC_START_MODE_MASK.

Example:

The synchronous spindle coupling at the beginning of the SERUPRO operation is retained for the part program start.

```
                                          ;  synchronous spindle coupling not
                                             configured
$MC_START_MODE_MASK = 'H400'              ;  is switched off
$MC_START_MODE_MASK_PRT = 'H00'           ;  remains active
$MC_ENABLE_START_MODE_MASK_PRT = 'H01'    ;  $MC_START_MODE_MASK_PRT is evaluated
                                             in SERUPRO instead of
                                             $MC_START_MODE_MASK
```

8.8.10 System variable

Determination of the SERUPRO status

The current SERUPRO status in the part program or synchronized action can be determined via the following system variables:

System variable	Meaning
$AC_ASUP, bit 20	ASUB activation reason: System ASUB, because the SERUPRO search type has reached the search target
$AC_SERUPRO	SERUPRO active
$P_ISTEST	Program test active; $P_ISTEST == 1 from the start of SERUPRO to the end
$P_SEARCHL	Last active block search type; $P_SEARCHL == 5 from the start of SERUPRO to reset or end of program

$AC_SERUPRO and $P_ISTEST

Note

During preprocessing with system variables $P_ISTEST and $AC_SERUPRO, a check is made as to whether the SERUPRO target block has already been found. If so, an implicit preprocessing stop is inserted before the system variables are evaluated. As a result, interpretation is halted and not continued again until SERUPRO is deactivated in the main run as well. The decision as to whether SERUPRO must be active or inactive is then made correctly.

8.9 Program operation

PLC, MD, operation

The execution of part programs can be controlled via the HMI in many ways using PLC inputs, machine data settings and operator inputs.

Definition

The execution of part programs or part program blocks in AUTOMATIC or MDA modes is referred to as program operation.

Channel control

Every channel can be manipulated by means of interface signals from the PLC. The control is exercised via mode groupspecific or channelspecific interface signals. An overview of these signals is given under data lists in this Description of Functions.

Status messages

Each channel reports its current program operation status to the PLC with interface signals. These signals are, in turn, divided up into mode groupspecific and channelspecific signals.

8.9.1 Initial settings

Machine data

Defined conditions can be set via machine data for the program operation or certain implementations of the NC language scope.

Initial settings

Initial settings can be programmed in channel-specific machine data for each channel. These initial settings affect, for example, G groups and auxiliary function output.

Auxiliary function output

The timing for output of auxiliary functions can be predefined via machine data AUXFU_x_SYNC_TYPE (MD22200, 22210, 22220, 22230, 22240, 22250, 22260), (output timing for M, S, T, H, F, D, E functions). For more detailed explanations, see Section "H2: Auxiliary function outputs to PLC (Page 395)".

G groups

An initial programming setting can be specified for each of the available G groups using MD20150 $MC_GCODE_RESET_VALUES (reset state of G groups). This initial setting is automatically active during program start or in Reset until it is deselected by a G command from the same G group.

Via the MD22510 $MC_GCODE_GROUPS_TO_PLC (G codes, which are output to interface NCK-PLC after block change / RESET), the output of the G codes to the PLC interface can be activated.

A list of G groups with the associated G functions is available in:

References:
Programming Manual, Fundamentals

Basic configurations of the NC language scope for SINUMERIK solution line

For SINUMERIK 840D sl, certain basic configurations of the NC language scope can be generated (configurable) via machine data. The options and functions of the NC language scope is specially tailored (configured) to the needs of the user.

NC language scope

The way that non-active options and functions should be moved with NC language commands can be set via the machine data MD10711 $MN_NC_LANGUAGE_CONFIGURATION:

0: All available language commands can be programmed. Whether or not the needed function is activated can only be recognized upon execution.

If only **certain options** are enabled and **not all operations** are available:

1: All **the language commands are known**. Language commands for non-enabled options are already recognized at the beginning of the program interpretation and lead to the alarm 12553 "option/function is not active."

2: Only those language commands are known which correspond to the **current scope of enabled options** of the NCK software. All commands for non-enabled options are not recognized and trigger the alarm 12550 "Name not defined or option/function not available".

Note

Option-free functions also have the status "enable option"

If only **certain functions are activated**:

3: All **the language commands are known**. Non-activated functions are already recognized at the beginning of the program interpretation and result in the alarm 12553 "Option/function is not active". For example, if the option date is set for the cylinder surface transformation, but the transformation is not activated in machine data MD24100 $MC_TRAOF_TYPE_1, then the programming of TRACYL triggers the alarm 12553.

4: Only those NC language commands are known which correspond to the **current scope of active functions** of the NCK software. All commands for non-active functions are not recognized and trigger the alarm 12550 "Name not defined or option/function not available". Whether the command in question is generally unavailable in the Siemens NC language or whether this is true only on the corresponding system cannot be distinguished in this scenario.

Whether the current NC language scope of enabled options and active functions is also truly programmable can be checked using the STRINGIS program command, see example.

Check sample application for NC language scope on cylinder surface transformation TRACYL

The cylinder surface transformation is optional and must be enabled beforehand. In order to check this, the following initial conditions are assumed:

The cylinder surface transformation option is **not** enabled and the machine data $MN_NC_LANGUAGE_CONFIGURATION = 2; NC language command TRACYL is unknown

The following program is started

```
Program code                              Comment
N1 R1=STRINGIS("TRACYL")                  ; R1 is 0 (TRACYL is an unknown name)
N2 IF STRINGIS("TRACYL")==204
N3 TRACYL(1, 2, 3)                        ; Block is not interpreted
N4 ELSE
N5 G00
N6 ENDIF
N7 M30
```

Example of whether STRINGIS result is programmable or not

The result of `STRINGIS` = number-coded return value (three-digit)

Number coding of the basic information (1st digit from the left):

000 Name is unknown, programming is denied with Alarm 12550

100: Name is known, but cannot be programmed, triggers alarm 12533

200: Name/symbol is known, but interpretation is not possible

2xx: Name/symbol is known, the **command can be programmed**, if xx > 0

Definition for name/symbol:

Name: Any STRING that is checked to see
whether it is a component of the NC language in the existing NCK version or configuration.

Symbol: Contains the description or significance of an NC language command that is needed for the NC program and cycle interpretation.

Dependent on machine data MD10711

`$MN_NC_LANGUAGE_CONFIGURATION` = (set value) results in the following interpretations of the option and function relative to their programmability **2**xx:

Table 8- 1 Setting options

MD10711 =		**0**	**1**	**2**	**3**	**4**
Option	Function	Return value as the basic information (1st digit from the left)				
0	0	**2**	1	0	1	0
1	0	**2**	**2**	**2**	1	0
1	**1**	**2**	**2**	**2**	**2**	**2**
0	1	**2**	1	0	1	0

Definition for option/function:

0 corresponds to option not activated or function deactivated
1 corresponds to option activated or function activated

For more detailed information on the value ranges of 2xx programmable functions, see:

References:
Programming Manual, Job Planning, Additional Functions,
Section: Check scope of NC language present (STRINGIS)

8.9.2 Selection and start of part program or part program block

Reset status

Channel status

A part program can be selected only if the relevant channel is in the Reset state.

Start command, channel status

There are two possible START commands for initiating processing of a part program or part program block:

- The channel-specific interface DB21, ... DBX7.1 (NC Start), which is usually controlled from the machine control panel key NC Start, starts program execution in the same channel.
- With the NC instruction START, program execution in the first channel can be started from the second channel, for example.

The START command can only be executed in AUTOMATIC and MDA modes. For this, the channel concerned must be in the following state:

DB21, ... DBX35.7(channel status reset) or

DB21, ... DBX35.6 (channel status interrupted).

Signals, Alarms

Required signal states

The part program can now be enabled for execution in the channel with the START command on the condition that certain signal states exist on the machine.

The following enable signals are relevant on the NC/PLC interface:

- DB11 DBX4.4 (mode group ready) must be present
- DB11 DBX0.7 (mode group reset) must not be present
- DB21, ... DBX1.7 (activate program test) must not be present
- DB21, ... DBX7.0 (NC start disable) must not be present
- DB21, ... DBX7.2 (NC stop at the block limit) must not be present
- DB21, ... DBX7.3 (NC stop) must not be present
- DB21, ... DBX7.4 (NC stop axes plus spindle) must not be present
- DB21, ... DBX7. 7 (reset) must not be present
- DB10 DBX56.1 (emergency stop) may not be present
- No axis or NCK alarm must be active

For a further explanation of the individual signals see Section 5.

Execution of command

The part program or the part program block is automatically executed and the the following interface signals are set:

DB21, ... DBX35.5 (channel status reset)

DB21, ... DBX35.0 (program status running)

The program is processed until the end of the program has been reached or the channel is interrupted or aborted by a STOP or RESET command.

Alarms

Under certain conditions the START command will have no effect and one of the following alarms will be triggered:

- 10200 "No NC Start permitted with active alarm"
- 10202 "No NC Start permitted with active command"
- 10203 "No NC Start permitted for non-referenced axes"

References:
Diagnostics Manual, Alarms

8.9.3 Part program interruption

Channel status

A part program interruption is only executed when the channel and program are active:

- DB21, ... D35.5 == 1 ("Channel active")
- DB21, ... D35.0 == 1 ("Program running")

STOP commands

The part program processing can be interrupted via the following STOP commands:

- DB21, ... DBX7.2 ("NC stop at the block limit)
- DB21, ... DBX7.3 ("NC stop")
- DB21, ... DBX7.4 ("NC stop, axes plus spindles")
- DB21, ... DBX2.0 ("Single block")
- Program command `M00` or `M01`

The channel and program are then in the state "interrupted":

- DB21, ... D35.6 == 1 ("Channel interrupted")
- DB21, ... D35.3 == 1 ("Program interrupted")

References

A detailed description of the interface signals can be found in:

Function Manual Basic Functions; NC/PLC Interface Signals (Z1)

Sequence

The following actions are performed after a STOP command:

- Interruption of the part program execution at the next block limit (with "NC stop at the block limit", M00, M01 or single block), or immediately for all other STOP commands.
- The traversing axes of the channel are stopped via braking ramp. The braking of the axes down to standstill can be extended over several blocks.
- The block indicator shows the current block at the point of interruption.
- The auxiliary functions that have not been output before the point of interruption are no longer output.

Possible actions in the interrupt state

Various functions can be performed in the channel during a part program interruption, for example:

- Overstore
  References
  Operating Manual, HMI Advanced, Section "Machine operating area" > Automatic mode" > "Overstore"
- Block search
  References
  Function Manual, Basic Functions; Section "Mode group, channel, program operation, reset response (K1)" > "Block search" or "Block search type 5 SERUPRO"
- Repositioning (REPOS)
  References Function Manual, Basic Functions; Section "Mode group, channel, program operation, reset response (K1)" > "Block search type 5 SERUPRO" > "REPOS" > "Repositioning with controlled REPOS"
- Oriented tool retraction
  References
  - Programming Manual, Job Planning; Section "Tool offsets"
  - Description of Functions, Basic Functions; Section "Tool offsets (W1)" > "Orientable toolholders" > ""
- Interrupt routine (see Section "Asynchronous subprograms (ASUBs), interrupt routines (Page 616)")

- DRF function, offset of the workpiece zero
  References
  Function Manual, Extended Functions; Manual and Handwheel Travel (H1)
- Starting the interrupted program via:
  - START command from another channel
    References
    Programming Manual, Job Planning; Section "Flexible NC programming" > "Program coordination (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM)"
  - NC/PLC interface:
    DB21, ... DBX7.1 (NC start)

8.9.4 RESET command

Command priority

Channel status

The RESET command can be executed in every channel state. This command is aborted by another command.

Commands

RESET-Command

The following Reset commands are available:

- DB11, ... DBX0.7 ("mode group reset")
- DB21, ... DBX7.7 ("Reset")

For a further explanation of the individual interface signals, please see
References: /FB1/ Function Manual Basic Functions; NC/PLC interface signals (Z1)

A RESET command can be used to interrupt an active part program or a part program block (in MDA).

After execution of the Reset command, the interface signal DB21, ... DBX35.7 ("Channel status Reset") is set.

The part program cannot be continued at the point of interruption. All the axes of the channel go into exact stop unless they are in followup mode. The same applies to the spindles configured in the channel.

The following actions are executed when the RESET command is triggered:

- Part program preparation is stopped immediately.
- Axes and, if they exist, spindles in the channel are decelerated along a braking ramp.
- Any auxiliary functions of the current block not yet output, are no longer output.
- The block indicator is reset to the beginning of the part program.
- All Reset alarms (channelspecific, axisspecific, spindlespecific) are cleared from the display.

8.9.5 Program status

The status of the selected program is displayed in the interface for each channel.
The PLC can then trigger certain responses and interlocks configured by the manufacturer depending on the status at the interface.
The program status is only displayed in AUTOMATIC and MDA mode. In all other modes the program status is aborted or interrupted.

Program states

The following program states are displayed at the NC/PLC interface (DB21, ...):

- DB21, ... DBX35.4 ("aborted")
- DB21, ... DBX35.3 ("interrupted")
- DB21, ... DBX35.2 ("stopped")
- DB21, ... DBX35.1 ("waiting")
- DB21, ... DBX35.0 ("running")

A detailed description of the interface signals can be found in Section "Signals from channel (DB21, ...) (Page 1867)".

Effects of commands and NC/PLC interface signals

The program state of an active program is influenced by various commands and NC/PLC interface signals. The following table shows the respective resulting state.

Initial state: "Running"					
Command or NC/PLC interface signal (IS)	Resulting state				
	Aborted	Interrupted	Stopped	Waiting	Running
IS "Reset"	x				
IS "NC stop"			x		
IS "NC stop at block limit"			x		
IS "NC stop axes and spindles"			x		
IS "Read-in disable"					x
IS "Feed stop, channelsp."					x
IS "Feed stop, axissp."					x
Feed override = 0%					x
IS "Spindle stop"					x
M02/M30 in a block	x				
M00/M01 in a block			x		
IS "Single block"			x		
IS "Delete distance-to-go"					x
Auxiliary functions output to PLC but not yet acknowledged			x		
Wait instruction in program				x	

8.9.6 Channel status

The current channel status is displayed in all operating modes at the NC/PLC interface (DB21, ...) for each channel.

Channel states

The following channel states are displayed at the NC/PLC interface (DB21, ...):

- DB21, ... DBX35.7 ("reset")
- DB21, ... DBX35.6 ("interrupted")
- DB21, ... DBX35.5 ("active")

A detailed description of the interface signals can be found in Section "Signals from channel (DB21, ...) (Page 1867)".

Effects of commands and NC/PLC interface signals

The channel state of an active program is influenced by various commands and NC/PLC interface signals. The following table shows the respective resulting state.

Initial state: "Active"			
Command or NC/PLC interface signal (IS)	Resulting state		
	Reset	Interrupted	Active
IS "Reset"	x		
IS "NC stop"		x	
IS "NC stop at block limit"		x	
IS "NC stop axes and spindles"		x	
IS "Read-in disable"			x
IS "Feed stop, channelsp."			x
IS "Feed stop, axissp."			x
Feed override = 0%			
IS "Spindle stop"			x
M02/M30 in a block	x		
M00/M01 in a block		x	
IS "Single block"		x	
IS "Delete distance-to-go"			x
Auxiliary functions output to PLC but not yet acknowledged			x
Wait instruction in program			x

The "Channel state active" signal is obtained when a part program or part program block is being executed or when axes are traversed in JOG mode.

8.9.7 Responses to operator or program actions

Status transitions

The following table shows the channel and program states that result after certain operator and program actions.

The left-hand side of the table shows the channel and program states and the mode groups from which the initial situation can be selected. Various operator/program actions are listed on the righthand side of the table, the number of the situation after the action has been carried out is shown in brackets after each action.

Table 8- 2 Responses to operator or program actions

Situation	Channel status			Program status					Active mode			Operator or program action (situation after the action)
	R	U	A	N	U	S	W	A	A	M	J	
1		x						x	x			RESET (4)
2		x						x		x		RESET (5)
3		x						x			x	RESET (6)
4	x			x					x			NC Start (13); Mode change (5 or 6)
5	x			x						x		NC Start (14); Mode change (4 or 6)
6	x			x							x	Direction key (15); Mode change (4 or 5)
7		x		x						x		NC Start (14)
8		x		x							x	NC Start (15)
9		x			x				x			NC Start (13); Mode change (10 or 11)
10		x			x					x		NC Start (16); Mode change (9 or 11)
11		x			x						x	Direction key (17); Mode change (9 or 10)
12		x				x			x			NC Start (13); Mode change (10 or 11)
13			x					x	x			NC Stop (12)
14			x	x						x		NC Stop (7); at block end (5)
15			x	x							x	NC Stop (8); at JOG end (6)
16			x		x					x		NC Stop (10); at block end (10)
17			x		x						x	NC Stop (11); at JOG end (11)
18			x				x		x			Reset (4); wait for other channel (18)

Channel status
R --> aborted
U --> interrupted
A --> running

Program status
N --> aborted
U --> interrupted
S --> stopped
W --> waiting
A --> running

Modes
A --> aborted
M --> aborted
J --> aborted

8.9.8 Part-Program Start

Start handling

Table 8- 3 Typical program sequence

Sequence	Command	Conditions (must be satisfied before the command)	Comments
1	Load program (via the operator interface or part program)		
2	Select AUTOMATIC mode		
3	Program preselection	Channel preselected Preselected channel in RESET state User ID sufficient for program preselection	
4	NC start for preselected channel	NC start disable not available Reference point approached in all axes	
5			Program execution
6	M02/M30/RESET	None	End of program

8.9.9 Example of a timing diagram for a program run

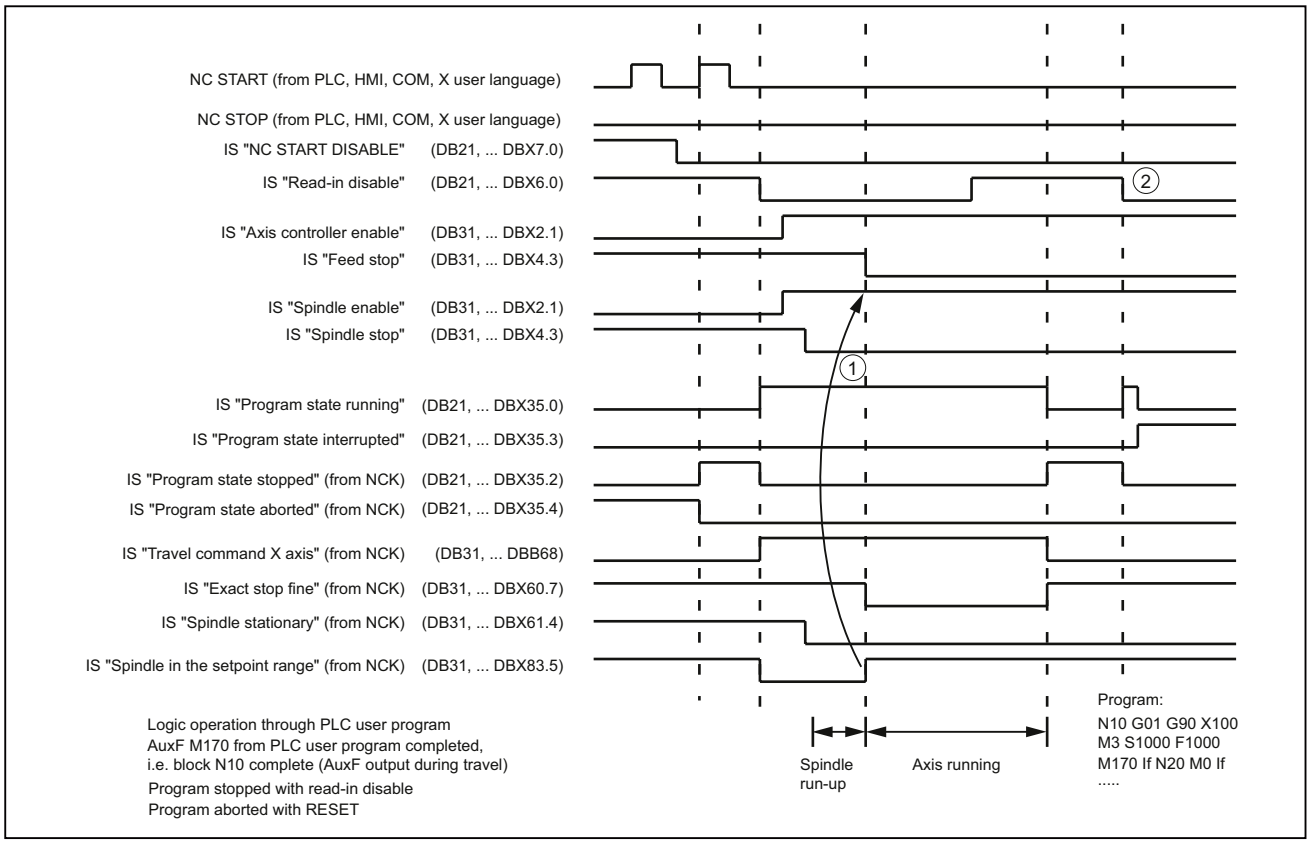


Figure 8-8 Examples of signals during the program run

8.9.10 Program jumps

8.9.10.1 Jump back to start of program

Function

With the function "Jump back to start of the program" the control jumps back from a part program to the beginning of the program. The program is then processed again.

As compared to the function "Program jumps to jump marks", with which a repeated processing of the program can also be implemented, the function "Jump back to the start of the program" offers the following advantages:

- The programming of a jump mark at the start of the program is not necessary.
- The program restart can be controlled through the NC/PLC interface signal:

  DB21, ... DBX384.0 (control program branching)
- The timer for the program runtime can be reset to "0" at the restart of the program.
- The timer for workpiece counting can be incremented by "1" at program restart.

Application

The function is used, if the processing of subsequent workpieces is to be done through an automatic program restart e.g. in case of turning machine with bar loader/-changer.

Activation

The jump back takes place only when the following NC/PLC interface signal is set:

DB21, ... DBX384.0 (control program branching) = 1

If the signal is at "0", then no jump back is exected and the program processing is continued with the next part program block after the function call.

Parameter assignment

Program runtime

The runtime of the selected NC program is stored in the system variable $AC_CYCLE_TIME. When starting a new program, the system variable is automatically reset to "0"(see Chapter " Program runtime (Page 697) ")

Via the following machine data it can be set that the system variable $AC_CYCLE_TIME is reset to "0" even in case of a program restart through the function "jump back to start of program":

MD27860 $MC_PROCESSTIMER_MODE (Activation of the program runtime measurement)

Bit	Value	Description
8	0	$AC_CYCLE_TIME is **not** reset to "0" by the function "jump back to start of program".
	1	$AC_CYCLE_TIME is reset to "0" by the function "jump back to start of program".

Note

In order that the setting of bit 8 can become effective, the measurement of the current program runtime must be active (MD27860 bit 1 = 1).

Workpiece count

After the part program end (M02 / M30) has been attained, the activated workpiece counters ($AC_TOTAL_PARTS / $AC_ACTUAL_PARTS / $AC_SPECIAL_PARTS) are incremented by "1" (see Chapter " Workpiece Counter (Page 705) ").

Via the following machine data it can be set that the activated workpiece counter is incremented even in case of a program restart through the function "jump back to start of program":

MD27880 $MC_PART_COUNTER (activation of workpiece counters)

Bit	Value	Description
		In case of a program restart through the function "jump back to start of program", the workpiece counter:
7	0	$AC_TOTAL_PARTS is **not** incremented.
	1	$AC_TOTAL_PARTS is incremented.
11	0	$AC_ACTUAL_PARTS is **not** incremented.
	1	$AC_ACTUAL_PARTS is incremented.
15	0	$AC_SPECIAL_PARTS is **not** incremented.
	1	$AC_SPECIAL_PARTS is incremented.

Programming

The function is called in the main or the part program via the command GOTOS.

Syntax: GOTOS

Parameters: none

Application in synchronized actions: not possible

Note

GOTOS internally initiates a STOPRE (pre-processing stop).

Example

Programming	Comment
N10 ...	; Beginning of the program
...	
N90 GOTOS	; Jump to beginning of the program
...	

8.9.11 Program section repetitions

8.9.11.1 Overview

Function

The program section repetition allows the repetition of any labeled section of a part program.

For more information on labels, please see:

References:

Programming Manual Fundamentals; Program Jumps and Program Repetitions

Definition options of part program sections

The program repetition offers various options for defining a part program section that is supposed to be repeated:

- A single part program block
- A part program section after a start label
- A part program section between a start label and end label
- A part program section between a start label and the key word: ENDLABEL

References:

Programming Manual, Job Planning; Section "Program coordination"

8.9.11.2 Individual part program block

Functionality

Via REPEATB (B=Block) in part program block N150, the part program processing branches to the part program block N120 that is labeled START_1. This is repeated x number of times. If P is not specified, the program section is repeated exactly once. After the last repetition, the part program is continued with the part program block N160 following the REPEATBinstruction.

```
:
N100 ...
N120 START_1: ...                    ; Label: START_1
N130 ...
N140 ...
N150 REPEATB START_1 P=n             ;  Repetition after: START_1
N160 ...
:
```

Note

Label search direction

The part program block identified by the label can appear before or after the REPEATB statement. The search initially commences toward the start of the program. If the label is not found, a search is made in the direction of the program end.

Programming

Syntax:

```
REPEATB <Label> [P=n]
```

Label	Start label to which the instruction: REPEAT branches Type: String
P	Number of repetitions
-{}-n	Number of repetitions Type: Integer

8.9.11.3 A part program section after a start label

Functionality

Via REPEAT in part program block N150, the part program processing branches to the part program block N120 that is labeled START_1. This part program block and all of the following part program blocks (N130 and N140) are repeated x number of times up to the part program block that contains the REPEATinstruction (N150). If P is not specified, the part program section (N120 - N140) is repeated exactly once. After the last repetition, the part program is continued with the part program block N160 following the REPEATinstruction.

```
:
N100 ...
N120 START_1: ...                    ; Start label: START_1
N130 ...
N140 ...
N150 REPEAT START_1 P=n              ;  Repetition after: START_1
N160 ...
:
```

Note

Label search direction

The part program block marked with the Start label **must** come before the REPEATinstruction.

Programming

Syntax: `REPEAT <Label> [P=n]`

Label	Start label to which the instruction: REPEAT branches Type: String
P	Number of repetitions
-{}-n	Number of repetitions Type: Integer

8.9.11.4 A part program section between a start label and end label

Functionality

Via REPEAT in part program block N160, the part program processing branches to the part program block N120 that is labeled START_1 with a start label. This part program block and all the part program blocks up to and including the part program block marked with the end label END_1 (N140) are repeated x number of times. If P is not specified, the part program section (N120 - N140) is repeated exactly once. After the last repetition, the part program is continued with the part program block N170 following the REPEATinstruction.

```
:
N100 ...
N120 START_1: ...                    ; Start label: START__1
N130 ...
N140 END_1 ...                       ; End label: END_1
N150 ...
N160 REPEAT START_1 END_1 P=n        ; Repetition: START_1 until END_1
N170 ...
:
```

Note

Label search direction

The program section marked with the Start and End labels can come before or after the REPEATinstruction. The search initially commences toward the start of the program. If the Start label is not found, a search is made in the direction of the program end.

If the REPEAT instruction is between the Start and End label, only the part program section from the Start label to the REPEAT instruction is repeated.

REPEAT instruction repeated.

Programming

Syntax: `REPEAT <Start_Label> <End_Label> [P=n]`

Start_Label	Start label to which the instruction: REPEAT branches. Beginning of the part program section that is repeated. Type: String
End_Label	End of the part program section that is repeated. Type: String
P	Number of repetitions
n	Number of repetitions Type: Integer

8.9.11.5 A part program section between a Start label and the key word: ENDLABEL

Functionality

Via REPEAT in part program block N150, the part program processing branches to the part program block N120 that is labeled START_1 with a start label. This part program block and all the part program blocks up to and including the part program block marked with the key word ENDLABEL (N140) are repeated x number of times. If P is not specified, the part program section (N120 - N140) is repeated exactly once. After the last repetition, the part program is continued with the part program block N170 following the REPEATinstruction.

```
:
N100 ...
N120 START_1: ...                    ; Start label: START__1
N130 ...
N140 ENDLABEL: ...                   ; End label: Keyword ENDLABEL
N150 ...
N160 REPEAT START_1 END_1 P=n        ; Repetition: START_1 until END_1
N170 ...
:
```

Note

Label search direction

The program section marked with the Start and End labels can come before or after the REPEATinstruction. The search initially commences toward the start of the program. If the Start label is not found, a search is made in the direction of the program end.

If no keyword ENDLABEL is located between the Start label and the REPEAT instruction, the part program section from the Start label to the REPEAT instruction is repeated.

Programming

Syntax:

```
REPEAT <Label> [P=n]
```

Label	Start label to which the instruction: REPEAT branches. Beginning of the part program section that is repeated. Type: String
P	Number of repetitions
n	Number of repetitions Type: Integer

8.9.12 Event-driven program calls

8.9.12.1 Function

What is the purpose of the function?

The function "Event-driven program calls" offers the possibility of letting an application program run implicitly during certain events, such as for making default settings of functions of initializations.

Events

Triggering events can be:

- Part program start
- Part program end
- Operator panel reset
- Power-up of the NC control

The triggering events are selected with the machine data MD20108 $MC_PROG_EVENT_MASK (see Section "Parameterization (Page 603)").

User program

In the default setting the program **_N_PROG_EVENT_SPF** is activated after the triggering event occurs. If a different application program is to be activated, then it must be entered in the machine date MD11620 $MN_PROG_EVENT_NAME (see Section "Parameterization (Page 603)").

The application program activated by the event is basically processed in the channel, in which the respective event occurred.

The application program is executed with the lowest priority and so can be interrupted by the user ASUB.

Processing sequence

Sequence during activation through part program start

Initial state:

Channel:	In the Reset state
Mode:	AUTO
	AUTO + overstoring or MDA
	TEACHIN

1. NC Start
2. Initialization sequence with evaluation of:

   MD20112 $MC_START_MODE_MASK (definition of the control default settings in case of NC START)
3. Implicit call of _N_PROG_EVENT_SPF as part program
4. Processing of the data part of the main program
5. Processing of the program part of the main program

Sequence during activation through part program end

Initial state:

Channel:	In active state
Mode:	AUTO
	AUTO + overstoring or MDA
	TEACHIN

1. Block with end of part program is changed
2. Control activates reset-sequence with evaluation of machine data:

   MD $MC_RESET_MODE_MASK

   $MC_GCODE_RESET_VALUES

   $MC_GCODE_RESET_MODE
3. Implicit call of _N_PROG_EVENT_SPF as ASUB
4. Control activates reset-sequence with evaluation of machine data:

   $MC_RESET_MODE_MASK

   $MC_GCODE_RESET_VALUES

   $MC_GCODE_RESET_MODE

   ⇒ The G code reset position continues to be set via machine data!

Sequence during activation through operator panel reset

Initial state:

Channel: Any
Mode: Any

1. Control activates reset-sequence with evaluation of machine data:
   MD $MC_RESET_MODE_MASK
   $MC_GCODE_RESET_VALUES
   $MC_GCODE_RESET_MODE
2. Implicit call of _N_PROG_EVENT_SPF as ASUB
3. Control activates reset-sequence with evaluation of machine data:
   $MC_RESET_MODE_MASK
   $MC_GCODE_RESET_VALUES
   $MC_GCODE_RESET_MODE
   ⇒ The G code reset position continues to be set via machine data!

Sequence during activation through power-up

1. Control activates after power-up reset-sequence with evaluation of machine data:
   MD $MC_RESET_MODE_MASK
   $MC_GCODE_RESET_VALUES
   $MC_GCODE_RESET_MODE
2. Implicit call of _N_PROG_EVENT_SPF as ASUB
3. Control activates reset-sequence with evaluation of machine data:
   $MC_RESET_MODE_MASK
   $MC_GCODE_RESET_VALUES
   $MC_GCODE_RESET_MODE
   ⇒ The G code reset position continues to be set via machine data!

Signal chart

The following diagrams show the signal chart of the NC/PLC interface signals DB21, ... DBB35 ("Program status" and "Channel status") in case of event-driven program call:

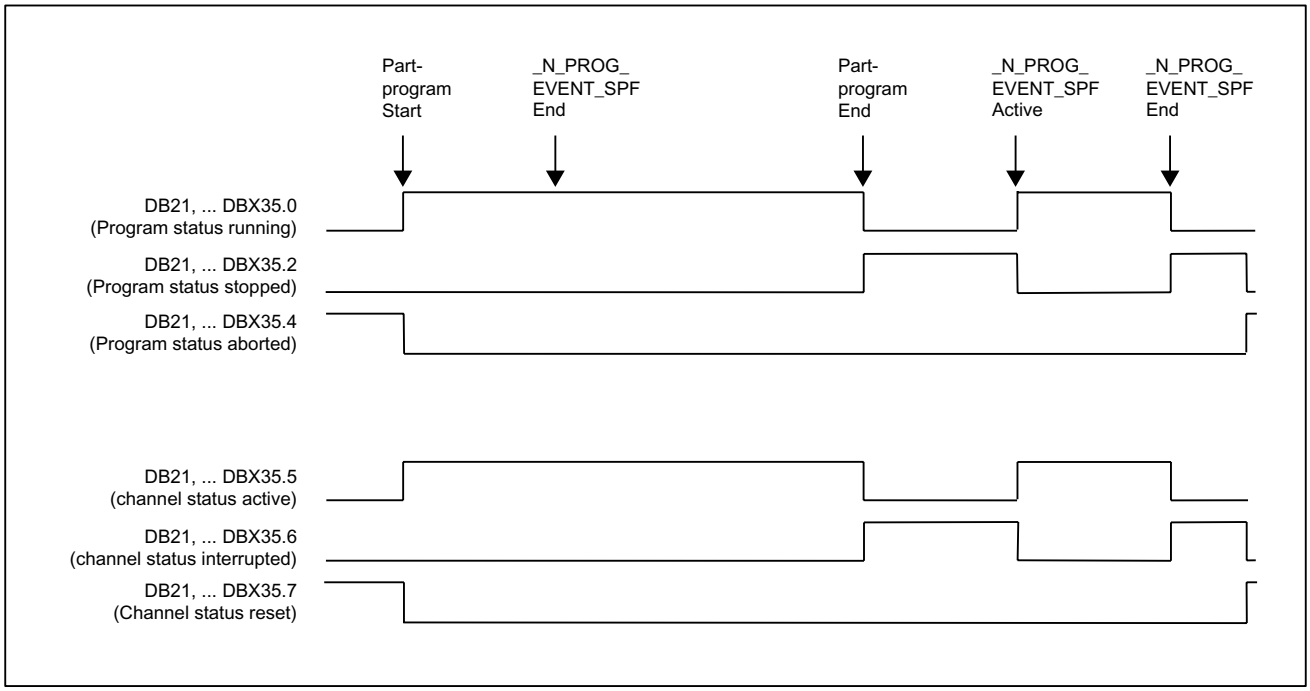


Figure 8-9 Signal chart in case of activation through part program start and part program end

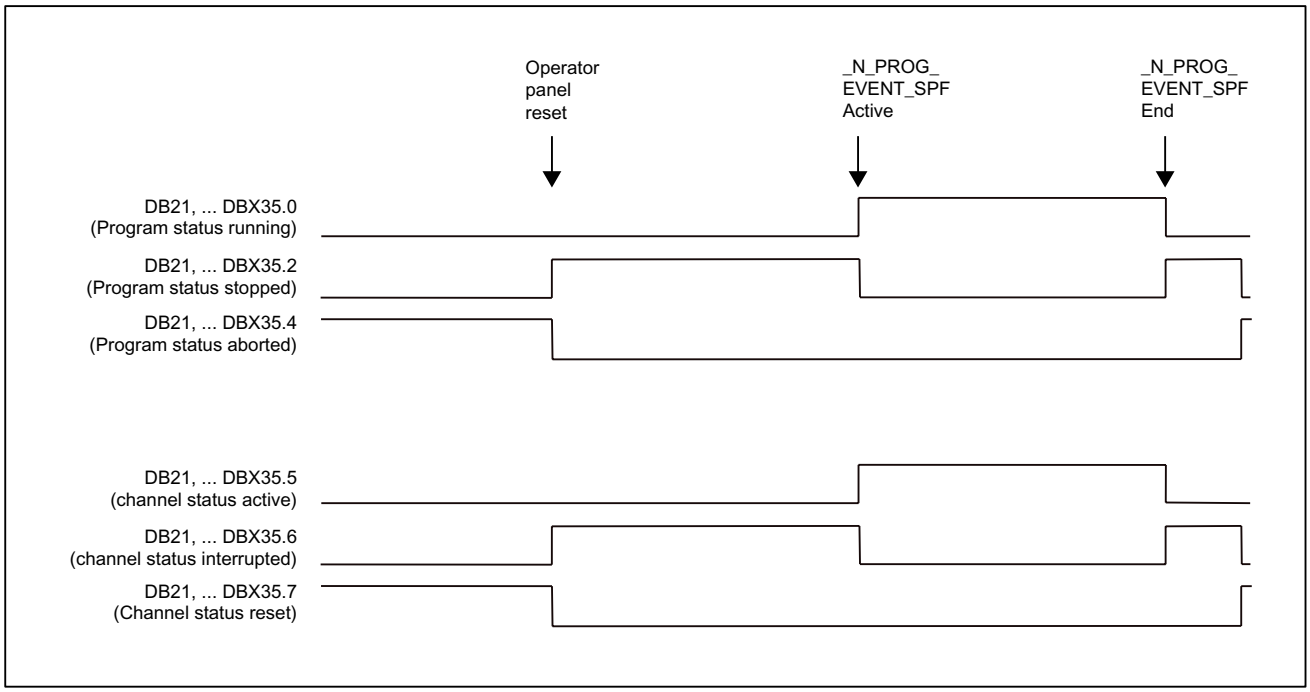


Figure 8-10 Signal chart during activation through operator panel reset

Note

DB21, ... DBX35.4 ("Program status aborted") and DB21, ... DBX35.7 ("Channel status reset") are only received if event-driven user program is complete. Between program end and the start of the event-driven application program these states are not imported. This is also the case between an operator panel reset and the start of the application program.

Display

The information about the triggering event is provided to the PLC via the NC/PLC interface byte DB21, ... DBB376.

Bit	Value	Meaning
0	1	Part program start from the channel status reset
1	1	Part program end
2	1	Operator panel reset
3	1	Power-up
4	1	1st start after block search (see "Automatic Start of an ASUB after block search (Page 528)")

The global query of DB21, ... DBB376 on 0 enables the determination, whether an event-driven application program is active at all.

If the event-driven application program has expired or has been interrupted with RESET, then the related display bit is deleted in the interface. For a very brief program, the corresponding bit remains for at least the duration of a complete PLC cycle.

8.9.12.2 Parameterization

Triggering event

Which events the application program should activate, is set channel-specifically in the machine data:

MD20108 $MC_PROG_EVENT_MASK (event-controlled program call)

Bit	Value	Meaning
0	1	Activation through **part program start**
1	1	Activation through **part program end**
2	1	Activation through **operator panel reset**
3	1	Activation through **run-up** of the NC control

Requirement for the activation:

The application program (default setting: _N_PROG_EVENT_SPF) must be loaded and cleared.

Note

MD20108 $MC_PROG_EVENT_MASK is ignored during the simulation.

User program

In the default setting after an event set with MD20108 occurs, the program **_N_PROG_EVENT_SPF** is activated from the directory _N_CMA_DIR.

If another program is to be activated, then the name of this user program must be entered in the following machine data:

MD11620 $MN_PROG_EVENT_NAME

The specified program must be present in one of the cycle directories.

The following search path is run when an event set with MD20108 occurs.

1. /_N_CUS_DIR/ for user cycles
2. /_N_CMA_DIR/ for manufacturer cycles
3. /_N_CUS_DIR/ for standard cycles

The first found program with the given name is called.

Note

The specified name is checked syntactically as in case of a subprogram identifier, i.e. the first two characters must be letters or underscores (no numbers). Prefix (_N_) and suffix (_SPF) of the program names are added automatically, if not specified.

Note

The same protection mechanisms that can be activated for cycles (protection levels for writing, reading, etc.) are activated.

Behavior when starting a user ASUB

The behavior of the function "event-driven program call" upon start of a user ASUB from the reset channel state can be set channel-specifically with the machine data:

MD20109 $MC_PROG_EVENT_MASK_PROPERTIES

Bit	Value	Meaning
0	0	The occurrence of an event set with MD20108 (part program start, part program end and/or operator panel reset) leads to the activation of the event-driven user program.
	1	The occurrence of an event set with MD20108 does **not** lead to the activation of the event-driven user program.

Behavior when the single-block processing is set

The behavior of the function "event-driven program call" in case of set single-block processing can be set channel-specific with the machine data:

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK

Bit	Value	Meaning
		In the event-driven user program:
0		• After an activation through part program start:
	0	The single-block processing is effective
	1	The single-block processing is suppressed
1		• After an activation through part program end:
	0	The single-block processing is effective
	1	The single-block processing is suppressed
2		• After an activation through operator panel reset:
	0	The single-block processing is effective
	1	The single-block processing is suppressed
3		• After an activation through run-up:
	0	The single-block processing is effective
	1	The single-block processing is suppressed

If the single-block processing is suppressed, then the event-driven user program is processed without interruption.

Note

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK affects all single-block processing types.

Note

The single-block processing in the event-driven user program can be switched-off through the following configuration:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop) bit 0 = 1

The differentiated settings in MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK are then ineffective.

Behavior when the read-in disable is set

The behavior of the function "event-driven program call" in case of set read-in disable (DB21, ... DBX6.1 = 1) can be set channel-specifically with the machine data:

MD20107 $MC_PROG_EVENT_IGN_INHIBIT

Bit	Value	Meaning
		In the event-driven user program:
0		• After an activation through part program start:
	0	The read-in disable is effective
	1	The read-in disable is suppressed
1		• After an activation through part program end:
	0	The read-in disable is effective
	1	The read-in disable is suppressed
2		• After an activation through operator panel reset:
	0	The read-in disable is effective
	1	The read-in disable is suppressed
3		• After an activation through run-up:
	0	The read-in disable is effective
	1	The read-in disable is suppressed

Note

For bit 0 = 1 (user program is activated after part program start) the following constraint is applicable:

If the user program is ended with the part program command RET, then RET always leads to an executable block (similar to M17).

In case of bit 0 = 0, RET interpretation is done in the Interpreter and leads to an executable block.

Suppress updating of the display of the program and channel states.

In order to avoid a flickering of the display of the program and the channel states in the operator panel, the updating of the display can be suppressed for the execution of the normally very brief event-driven user program. In the display, the program and channel states remains visible before activation of the event-driven user program (e.g. "program state canceled" and "channel state reset").

The parameterization of this function is performed with the channel-specific machine data:

MD20192 $MC_PROG_EVENT_IGN_PROG_STATE

Bit	Value	Meaning
		During execution of an event-driven user program, the updating of the display of the program and channel states:
1		• After an activation through part program end:
	0	Not suppressed
	1	Suppressed
2		• After an activation through operator panel reset:
	0	Not suppressed
	1	Suppressed
3		• After an activation through run-up:
	0	Not suppressed
	1	Suppressed

Note

The system variables $AC_STAT and $AC_PROG are not affected by this function, i.e. in the running event-driven user program, $AC_STAT is set to "active" and $AC_PROG to "running".

NC/PLC interface signals DB21, ... DBX35.0-7 ("Program state ..." and "Channel state ...") also remain unaffected.

Behavior for NC Stop

The behavior of the function "event-driven program call" for NC Stop (i.e. NC/PLC interface signal DB21, ... DBX7.2, 7.3 or 7.4 is set) can be set channel-specifically for the triggering event part program end, operator panel reset and run-up with the machine data:

MD20193 $MC_PROG_EVENT_IGN_STOP

Bit	Value	Meaning
		The event-driven user program is:
1		• After an activation through part program end:
	0	Stopped/prevented at NC stop
	1	Despite NC Stop been completely processed
2		• After an activation through operator panel reset:
	0	Stopped/prevented at NC stop
	1	Despite NC Stop been completely processed
3		• After an activation through run-up:
	0	Stopped/prevented at NC stop
	1	Despite NC Stop been completely processed

In this way, an edge change of the interface signal DB21, ... DBX7.3 (NC Stop) initiated by the user by activating the NC Stop key in case of reset or run-up is ignored during the execution of the event-driven user program and an undesired stop behavior at the machine is prevented.

Note

A programming of DELAYFSTON/ DELAYFSTOF in the event-driven user program cannot be provided with the behavior set with MD20193, because the NC Stop can cause an interruption before the execution of the first command DELAYFSTON.

8.9.12.3 Programming

User program

End of program

The following must be kept in mind, if the user program is to be activated through the part program start.

- The user program must be ended with M17 or RET.
- A jump back by means of REPOS command is not permitted and leads to an alarm.

Block display

The display can be suppressed in the current block display using the DISPLOF attribute in the PROC statement.

Processing status

Via the user M function the PLC can be informed about the processing status of the event-driven user program.

Scan for triggering event

The event, which causes the activation of the user program, can be queried in the user program with the following system variable:

$P_PROG_EVENT (event-driven program call active)

Value	Description
1	Activation through **part program start**
2	Activation through **part program end**
3	Activation through **Operator panel reset**
4	Activation through **Power up**
5	Activation after output of the last action block after **Block search** (see "Automatic Start of an ASUB after block search (Page 528)")

Query of the current channel

The application program is basically processed in the channel, in which the corresponding event has occurred. The current channel is queried in the user program with the following system variables:

$P_CHANNO (query of the current channel number)

Note

Power up is an event that takes place in all channels.

8.9.12.4 Boundary conditions

Emergency stop / error message

If an error is present when the operator panel is reset or after powerup EMERGENCY STOP or Mode group/NCKContinue, then the event-driven user program will only be processed after EMERGENCY STOP or the error has been acknowledged in all channels.

Note

The power up event occurs in all channels at the same time.

8.9.12.5 Examples

Example 1: Call of all events set with MD20108

Parameter assignment:

MD20108 $MC_PROG_EVENT_MASK = 'H0F'

Call of _N_PROG_EVENT_SPF for:

- Part program start
- Part program end
- Operator panel reset
- Ramp-Up

Programming:

```
Program code                                          Comment
PROC PROG_EVENT DISPLOF

IF ($P_PROG_EVENT==1)                                 ; Processing for part program start.
   MY_GUD_VAR=0                                       ; Initialize GUD variable
   RET
ENDIF

IF ($P_PROG_EVENT==2) OR ($P_PROG_EVENT==3)           ; Processing for part program end and
                                                        operator panel reset.
   DRFOF                                              ; Deactivate DRF offsets
   IF $MC_CHAN_NAME=="CHAN1"
      CANCEL(2)                                       ; Delete modal synchronized action 2
   ENDIF
   RET
ENDIF
```

```
Program code                                                    Comment
IF ($P_PROG_EVENT==4)                                           ; Sequence for power-up
    IF $MC_CHAN_NAME=="CHAN1"
        IDS=1 EVERY $A_INA[1]>5.0 DO $A_OUT[1]=1
    ENDIF
    RET
ENDIF

RET
```

Example 2: Call through Operator panel reset

Parameter assignment:

MD20108 $MC_PROG_EVENT_MASK = 'H04'

Call of _N_PROG_EVENT_SPF for:

- Operator panel reset

Programming:

```
Program code                        Comment
PROC PROG_EVENT DISPLOF
N10 DRFOF                           ; Deactivate DRF offsets
N20 M17
```

Example 3: Initialization of the function

Section of the startup file (_N_INITIAL_INI):

```
Program code                                    Comment
...
CHANDATA(3)                                     ; Initialization for channel 3
$MC_PROG_EVENT_IGN_INHIBIT='H04F'
$MC_PROG_EVENT_MASK='H04'
...
```

Meaning:

The part program _N_PROG_EVENT_SPF from the directory _N_CMA_DIR should be started automatically with the RESET key and processed till the end, regardless of whether the read-in disable is activated or deactivated.

8.9.13 Influencing the Stop events through Stop delay area

Stop delay area

The reaction to a stop event can be influenced by the conditioned interruptible area in the current part program. Such a program area is called stop delay area.

Within the stop delay areas there should be no stop and the feed should not be changed. Stops do not take effect until the program section has been completed (Example: making of a thread).

A stop delay area is defined with the part program commands:

`DELAYFSTON`	Start of a stop delay area
`DELAYFSTOF`	End of a stop delay area

References:
Programming Manual, Job Planning

Stop events

Overview of the NCK events that cause a stop:

NCK events	Reaction	Stop criteria
Reset and mode group RESET	Immediate	IS: DB21, ... DBX7.7 and DB11 DBX20.7
PROG_END	Alarm 16954	NC prog.: M30
Interrupt	Delayed	IS: "FC-9" and ASUP DB10 DBB1
DELDISTOGO_SYNC	Immediate	IS: "Delete distance-to-go" DB21, ... DBX6.2 and axial
PROGRESETREPEAT	Delayed	IS: "Clear number of subprogram passes" DB21, ... DBX6.3
PROGCANCELSUB	Delayed	IS: "Program level abort" DB21, ... DBX6.4
SINGLEBLOCKSTOP	Delayed	In the stop delay area: NC stops at the end of the 1st block outside the stop delay area. Single block is active before the stop delay area: IS: "NC Stop at block limit" DB21, ... DBX7.2
SINGLEBLOCK_IPO	Delayed	IS: "Activate single-block type 1" DB11 DBX21.7
SINGLEBLOCK_DECODIER	Delayed	IS: "Activate single-block type 2" DB11 DBX21.6
STOPALL	Immediate	IS: DB21, ... DBX7.4 and DB11 DBX20.6
STOPPROG	Delayed	IS: DB21, ... DBX7.3 and DB11 DBX20.5
OVERSTORE_BUFFER_END_REACHED	Alarm 16954	NC prog.: Stop because of empty overstore buffer
PREP_STOP	Alarm 16954	NC prog.: STOPRE and all implicit Stopres
PROG_STOP	Alarm 16954	NC prog.: M0 and M1
STOPPROGATBLOCKEND	Delayed	IS: "NC Stop at block limit" DB21, ... DBX7.2
STOPPROGATSUPEND	System fault	Subprogram end should always deselect the stop delay area.
WAITM	Alarm 16954	NC prog.: WAITM
WAITE	Alarm 16954	NC prog.: WAITE

NCK events	Reaction	Stop criteria
INIT_SYNC	Alarm 16954	NC prog.: INIT with parameter "S"
MMCCMD	Alarm 16954	NC prog.: MMC(STRING, CHAR)
PROGMODESLASHON	Delayed	IS: DB21, ... DBB26 Activate / switch over skip block
PROGMODESLASHOFF	Delayed	IS: DB21, ... DBB26 Deactivate skip block
PROGMODEDRYRUNON	Delayed	IS: DB21, ... DBX0.6 Activate DryRun
PROGMODEDRYRUNOFF	Delayed	IS: DB21, ... DBX0.6 Deactivate DryRun
BLOCKREADINHIBIT_ON	Delayed	IS: DB21, ... DBX6.1 Activate read-in disable
STOPATEND_ALARM	Immediate	Alarm: Alarm configuration STOPATENDBYALARM
STOP_ALARM	Immediate	Alarm: Alarm configuration STOPBYALARM
STOPATIPOBUFFER_ISEMPTY_ALARM	Immediate	Internal: Stop after alarm on empty IPO buffer
STOPATIPOBUF_EMPTY_ALARM_REORG	Immediate	Internal: Stop after alarm on empty IPO buffer
RETREAT_MOVE_THREAD	Alarm 16954	NC prog.: Alarm 16954 at LFON (Stop & fast lift in G33 not possible
WAITMC	Alarm 16954	NC prog.: WAITMC
NEWCONF_PREP_STOP	Alarm 16954	NC prog.: NEWCONF
BLOCKSEARCHRUN_NEWCONF	Alarm 16954	NC prog.: NEWCONF
SET_USER_DATA	Delayed	OPI: PI "_N_SETUDT"
ESR	Delayed	Extended stop and retract
EXT_ZERO_POINT	Delayed	External zero offset
STOPRUN	Alarm 16955	OPI: PI "_N_FINDST" STOPRUN

Reaction

The reaction to a stop event can be:

- **Immediate**

  Stops immediately even in stop delay area. Is known as a "hard stop event".

- **Delayed**

  Does not stop (even short-term) until after stop delay area. Is known as a "soft stop event".

- **Alarm 16954**

  Program is aborted because illegal program commands have been used in stop delay area.

- **Alarm 16955**

  Program is continued, an illegal action has taken place in the stop delay area.

- **Alarm 16957**

  The program area (stop delay area) enclosed by DELAYFSTON and DELAYFSTOF could not be activated. As a result, every stop will take effect immediately and is not subject to a delay! This will always occur when the deceleration begins before the stop delay area but ends within the stop delay area. Likewise, if the stop delay area is entered with an override of 0, the stop delay area also cannot be activated. (Example: A G4 before the stop delay area allows the user to reduce the override to 0. The next block in the stop delay area then begins with override 0 and the described alarm situation occurs.)

Note

MD11411 $MN_ENABLE_ALARM_MASK (activation of warnings) Bit 7 activates this alarm.

Stop criteria

A stop event can be triggered by the following

- NC/PLC interface signals from the PLC → "Hard" stop event
- Alarms with NOREADY response → "Hard" stop event
- Stop key → "Soft" stop event
- Read-in disable → "Soft" stop event
- Single block → "Soft" stop event

Note

Some NCK events are stopped for a short time, in order to perform a switching operation, and restart immediately. These include, e.g. the ASUB that stops the contour briefly in order to then start the ASUB program immediately. These events are also allowed in the stop delay area, however they are pushed back to its end and are thus considered "soft stop events".

Conditions

The following conditions apply while a stop delay area is being processed:

- A change in the **feed** is ignored while in the stop delay area. A **feed disable** is thus not effective until the program area has been exited, and is stopped.
- None of the main run axes, such as command axes and positioning axes, which are traversed with `POSA`, are stopped.
- Part program command `G4` is permitted in the stop delay area.

  Other part program commands that cause a stop in the meantime (e.g. `WAITM`), are not permitted and trigger the alarm 16954.
- A stop delay area entered with an override of 0% will not be accepted!

8.10 Asynchronous subprograms (ASUBs), interrupt routines

8.10.1 Function

8.10.1.1 General functionality

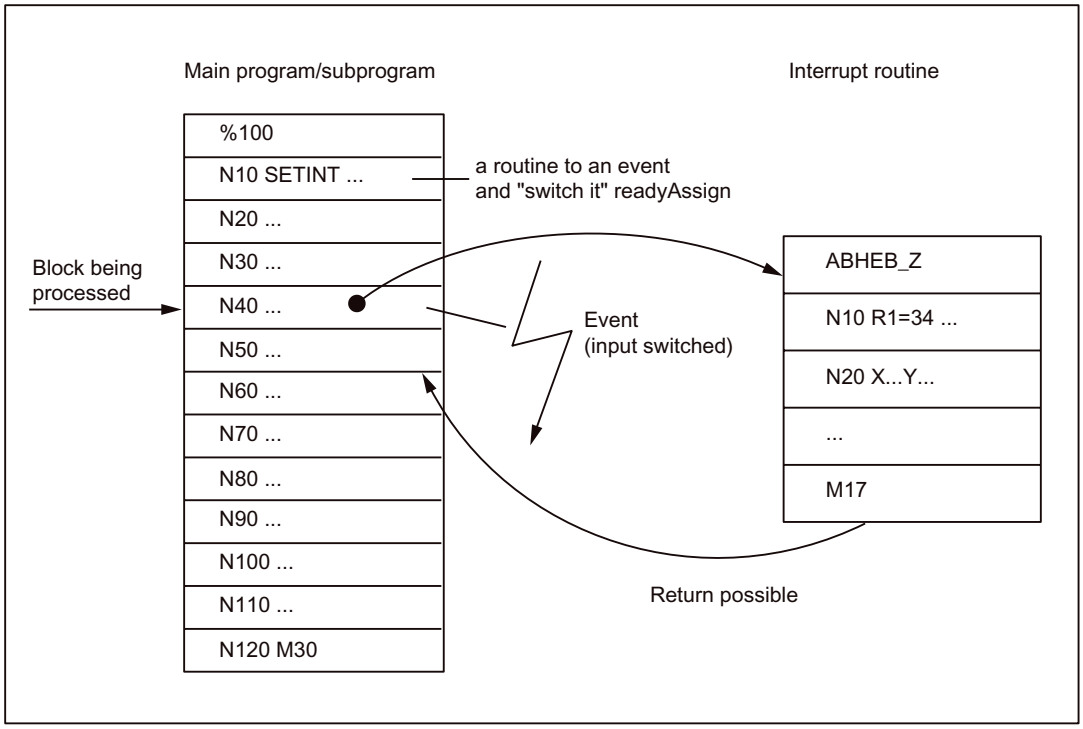
Note

The terms "asynchronous subprogram (ASUB)" and "interrupt routine" are used interchangeably in the description below to refer to the same functionality.

interrupt routines

Interrupt routines are normal part programs, which are started by interrupt events (interrupt inputs, process or machine status) related to the machining process or the relevant machine status.

Any part program block currently being executed will be interrupted by the routine if it is not specifically declared to be locked against interruption. It is possible to continue the part program at the point of interruption at a later stage.



Definition of interrupt routines

The command SETINT or an interrupt signal via the PI service "ASUB" must be assigned to a part program, which is supposed to act as interrupt routine. This turns the part program into an interrupt routine.

Interrupt signals

- A total of 8 interrupt signals (inputs) are available.
- All inputs can be controlled via the PLC.
- The first four interrupt signals are also controlled via the 4 rapid NC inputs of the NCU module.
- The signal status of the rapid NC inputs can be read out via the PLC interface (DB10).
- The transmission of the rapid NC input signals to the interrupt signals can be disabled via the PLC interface (DB10).

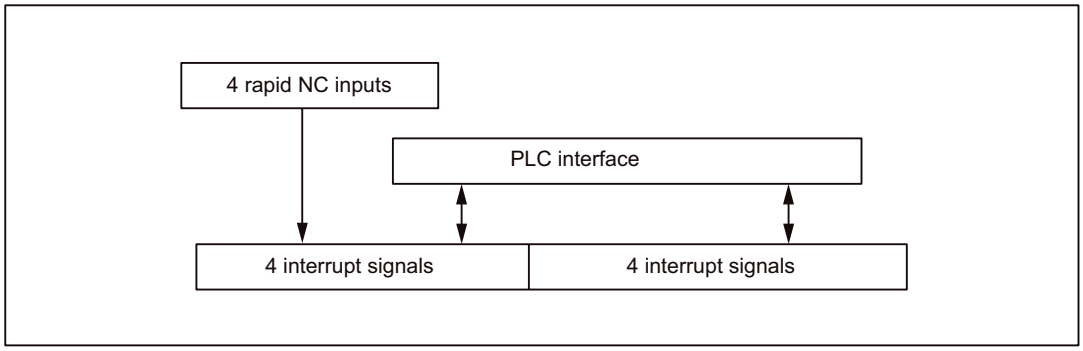


Figure 8-11 Interrupt signals

For further information about PLC control of the rapid NC inputs (interrupt signals) see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)".

References:
Function Manual, Extended Function; Digital and Analog NCK I/O (A4)

Call of interrupt routines

During program operation

It is possible to call the interrupt routines, when the mode groups are present in program operation. This means that the processing is done either in the mode type AUTOMATIC or MDA part program blocks.

Outside the program operation

Interrupt routines can also be activated in the following program states or mode types:

- JOG, JOG REF
- MDA Teach In, MDA Teach In REF, MDA Teach In JOG, MDA REF, MDA JOG
- AUTOMATIC, stopped, ready
- Not referenced

If an interrupt routine is activated in JOG or REF mode, it will interrupt any jogging and referencing operations in progress.

Activation

The activation of an interrupt routine can be initiated:

- By a 0/1 transition of the interrupt signal, triggered by a 0/1 transition at the rapid NC input
- By calling the "Function call ASUB" (see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)")
- By setting an output via synchronized action which indirectly sets an interrupt input via short-circuit (see "Examples (Page 629)").

  References:
  Function Manual, Synchronized Actions

Display

The activation of an interrupt routine is shown with the following NC/PLC interface signal:

DB21, … DBX378.0 (ASUB active)

8.10.1.2 Sequence of an interrupt routine in program operation

Decelerating the axes

Upon activation, all machine axes are decelerated to a standstill according to the acceleration ramp (MD32300 $MA_MAX_AX_ACCEL), and the axis positions are stored.

Reorganization

In addition to decelerating the axes, the previously decoded calculation blocks are calculated back to the interruption block, i.e. all the variables, frames and G codes are assigned the value that they would have at the point of interruption if the part program had not been previously decoded. These values are transferred to the buffer so that they can be called up again when the interrupt routine is completed.

Exceptions where no reorganization is possible:

- In thread cutting blocks
- With complex geometries (e.g. spline or radius compensation)

Processing of interrupt routine

The interrupt routine is automatically started on completion of reorganization.

The system handles the interrupt routine like a normal part program (nesting depth, etc.)

End of interrupt routine

After the end identifier (`M02`, `M30`, `M17`) of the interrupt routine has been processed, the axis traverses by default to the end position programmed in the part program block following the interruption block.

A `REPOS` instruction must have been programmed at the end of the interrupt routine if return positioning to the point of interruption is required, e.g.

N104 REPOSL M17

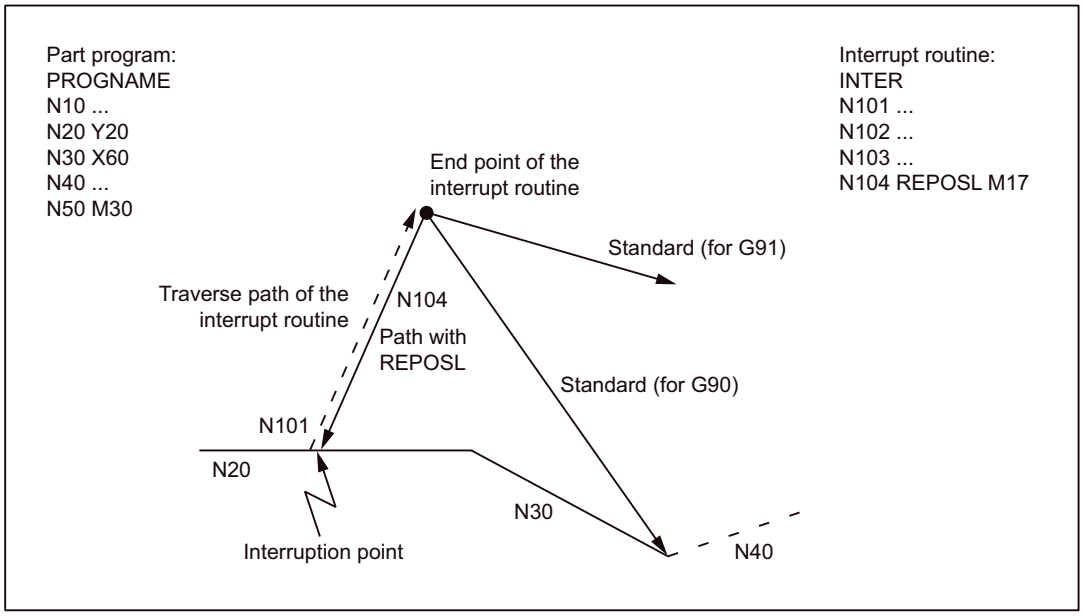


Figure 8-12 End of interrupt routine

8.10.1.3 Interrupt routine with REPOSA

If an interrupt routine with `REPOSA` triggered by the PLC (block FC9) "interrupted" in channel status in program operation is completed, then the following sequence is typical:

1. Before the re-approach to the contour, the controller stops and goes to program status "Stopped". The following NC/PLC-interface signal is set:

   DB21, ... DBX318.0 (ASUB is stopped)

2. The operator presses the START key. Thereupon, the signal DB21, ... DBX318.0 is reset and the re-approach motion starts.
3. At the end of the re-approach motion, the FC9 signal "ASUB done" is set and the path of the interrupted part program is continued.

Note

The NC/PLC-interface signal DB21, ... DBX318.0 (ASUB is stopped) is available only for the following case: Interrupt "interrupted" in program operation in the channel status.

Note

In case of interrupt routines that close without REPOS, the signals "Asub-Done" and DB21, ... DBX318.0 (ASUB is stopped) occur at the same time.

8.10.1.4 NC response

The different reactions of the control to an activated interrupt routine in the various operating states are given in the following table:

Status of NC	ASUB start	Control system reaction
Program is active	Interrupt, (PLC)	1. Fast retraction or stop axes 2. Interrupt the program for the duration of the ASUB 3. Approach of the interruption point, if REPOS in ASUB 4. Continuation of the part program
RESET	Interrupt, (PLC)	The ASUB is executed like a main program. RESET (without M30) is executed at the end of the ASUB. The next control system status depends on the following machine data: MD20110 $MC_RESET_MODE_MASK MD20112 $MC_START_MODE_MASK **References:** Function Manual Basic Functions; Axes, Coordinate Systems, Frames (K2), Chapter: "Workpiece-related actual-value system"
Program operation (AUTOMATC or MDA) + channel stopped	Interrupt, (PLC)	ASUB is executed. At the end of the ASUB the STOP state is reapplied. If REPOS in the ASUB: • The ASUB processing is stopped before the approach block. • The approach movement can be initiated with the Start key.
	Start key	Once the ASUB has been executed, processing of the interrupted program is resumed.
Manual mode + channel stopped	Interrupt, (PLC)	Control system assumes the status "internal program execution mode" for the addressed channel (not evident externally) and then activates the ASUB. The selected operating mode remains valid. The original status is resumed after execution of the ASUB (M17).
JOG AUTO Teach-In AUTO Teach reference pnt.	Interrupt, (PLC)	Stop processing, evaluate: MD11602 $MN_ASUP_START_MASK MD11604 $MN_ASUP_START_PRIO_LEVEL Internal switchover to "internal program execution mode" if appropriate, activate ASUB, restore status prior to ASUB start. Any LIFTFAST defined with SETINT is not activated in JOG mode.
MDA JOG, MDA Teach-In, MDA Teach reference pnt.	Interrupt, (PLC)	
Manual mode + channel running	Interrupt, (PLC)	The current active motion is stopped. The distancetogo is deleted. The remaining sequence of operations is the same as for "Manual mode, channel stopped".
Processing of INITIAL.INI	not possible	The signal "Interrupt request not possible" is generated.
Block search		
Alarm that cannot be removed by NC start.		
Digitalizing active		
Channel in fault condition		

8.10.2 Parameterization

Effect of mode group-specific signals

The effect of the mode group-specific signals (NC/PLC interface signals DB11) on channels of the mode group that is currently processing the interrupt routines, is set in the machine data:

MD11600 $MN_BAG_MASK.bit n = <value>

Bit	Value	Meaning
0	0	The mode group-specific signals (NC/PLC interface signals DB11) are effective.
	1	The mode group-specific signals (NC/PLC interface signals DB11) are not effective.

MD11600 also controls whether the internal program execution mode is imported only for the channel in which the interrupt routine has been activated, or to all channels in the mode group.

Bit	Value	Meaning
1	0	A mode group switchover is performed in all channels of the mode group.
	1	An internal mode group switchover is only performed in the channel in which an interrupt routine is active (only possible with bit 0==1).

If, because of the machine data setting, the channel in which the interrupt is being processed has left the mode group, the mode group signals "Mode group reset", "Mode group stop", etc. have **no** effect on this channel. In this way, the interrupt routine is run without being disturbed by the mode group signals.

Parameterizable start enable

In the basic setting, the start of an ASUB is blocked by the following states in the channel:

- Stop through NC stop, `M0` or `M01`
- Not all axes in the channel are referenced
- Read-in disable is active (DB21, ... DBX6.1 = 1)

The start can be enabled for specific states via the following machine data.

NC-specific start enable for NC stop, M0, M01, read-in disable

MD11602 $MN_ASUP_START_MASK

Channel-specific start enable for non-referenced axes in the channel

- MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK

  The start enable can be set separately for the following states via the machine data for event-driven program calls (ProgEvent) for non-referenced axes in the channel:

  – Part program start from channel state "Reset"
  – Part program end
  – Reset
  – Power On
  – First start after search

- MD20115 $MC_IGNORE_REFP_LOCK_ASUP

  The start enable can be issued interrupt-specifically via the machine data for the ASUB with non-referenced axes in the channel.

Manual start enable

If an ASUB cannot be started automatically because of the parameterized start enables, the user can manually issue a start enable by triggering **NC start**

Note

The ASUB for "fast retraction from the contour" (`LIFTFAST`) is started in every case.

Manual traversing during an ASUB interruption in JOG mode

Axes can be manually traversed by the operator using the traversing keys during the interruption of an ASUB started automatically in JOG mode. The function is enabled via:

MD11602 $MN_ASUP_START_MASK.bit 3

Note

Multi-channel systems

In multi-channel systems, the following machine data must also be set:

MD11600 $MN_BAG_MASK, bit 1 = 1

Continuation of the ASUB

After manually traversing the axes, NC start must be initiated by the operator. The axes are automatically traversed to the interruption point (REPOS). The ASUB is then continued at the interruption point.

Application example

In case of a single-slide turning machine a stock removal cycle is started as ASUB in the mode type JOG and with this a shaft several meters long is machined. During machining it is necessary to change the cutting edge of the tool. The machine operator stops the ASUB and retracts the axes manually using the traversing keys in order to change the cutting edge of the tool. After changing the cutting edge, the operator activates NC start. Repositioning at the interruption point is performed with the REPOS operation. Thereafter, the ASUB is continued.

Supplementary condition

Manual traversing of axes during the interruption of an ASUB in JOG mode is only possible if the ASUB has been activated from the program state "Aborted", channel state "Reset".

Effectiveness of the parameterized start enables

The user-specific interrupt signals are assigned the priorities 1 - 8, with 1 = highest priority (see Section "Programming (Page 626)"). Starting from the highest, up to which priority the parameterized start enables apply for the associated ASUB is defined with the following machine data:

MD11604 $MN_ASUP_START_PRIO_LEVEL = <priority>

Behavior with read-in disable

The following channel-specific machine data is used to set for each interrupt signal whether the assigned ASUB is processed despite a set read-in disable (DB21, ... DBX6.1 = 1), or whether the read-in disable is to apply:

MD20116 $MC_IGNORE_INHIBIT_ASUP, bit 0 - bit 31

Bit x is assigned to the interrupt signal (x+1).

Supplementary conditions

The settings in MD20116 $MC_IGNORE_INHIBIT_ASUP have no effect if the read-in disable in the ASUB is always to be ignored:

MD11602 $MN_ASUP_START_MASK, bit 2 = 1

Behavior when the single-block processing is set

The following channel-specific machine data is used to set for each interrupt signal whether the assigned ASUBs are processed without interruption with active single-block processing or whether the single-block processing is to apply:

MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP, bit 0 - bit 31

Bit x is assigned to the interrupt signal (x+1).

Supplementary conditions

- MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP only applies for IPO single block (SBL1).
- The settings in MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP have no effect if the single-block processing in the ASUB is always to be ignored:

  MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, bit 1 = 1

Suppressing the updating of the "Program and channel state" display

To avoid a flickering of the display of the program and the channel states on the user interface when executing a very short ASUB, the updating of the display can be suppressed. The program and channel state before activation of the ASUB is then displayed. The parameter assignment is performed via channel-specific machine data:

MD20191 $MC_IGN_PROG_STATE_ASUP, bit 0 - bit 31

Bit x is assigned to the interrupt signal (x+1).

The following NC/PLC interface signal is set when executing an ASUB with suppressed display:

DB21, … DBX378.1 == 1 (still ASUB active)

Special system variables and NC/PLC interface signals

The system variables and NC/PLC interface signals for the program state and the channel state are not affected by the suppression of the display during the execution of an ASUB:

- $AC_STAT (channel state)
- $AC_PROG (program state)
- DB21, ... DBX35.0 - 4 (program state)
- DB21, ... DBX35.5 - 7 (channel state)

Fast retraction from contour (LIFTFAST)

The following machine data can be used to set whether the retraction direction for "Fast retraction from the contour" (LIFTFAST) is to be mirrored when the "Mirror" function is active:

MD21202 $MC_LIFTFAST_WITH_MIRROR (fast retraction with mirroring)

The mirroring of the retraction direction only refers to the direction components perpendicular to the tool direction.

References

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

Lists Manual "Detailed Description of the Machines"

8.10.3 Programming

Assignment interrupt signal ↔ part program

The assignment interrupt signal ↔ part program is performed with the command SETINT.

Example:

```
Program code                                Comment
...
N20 SETINT(3) ABHEBEN_Z                     ; If input 3 switches, then interrupt routine
                                              "ABHEB_Z" should start.
...
```

Together with SETINT additionally the following commands can be programmed:

- LIFTFAST

  When the interrupt signal arrives, a "Fast retraction of the tool from the contour" is executed before the interrupt routine starts. The motion direction for the fast retraction is specified by the program instruction ALF.

- BLSYNC

  Upon receiving the interrupt signal, the current program block is processed and only then is the interrupt routine started.

Note

The assignment interrupt signal ↔ part program is cleared when the following happens:

- Channel in Reset state
- CLRINT instruction in part program

Priorities

If several SETINT instructions are in the part program and therefore several signals can be simultaneously received, the assigned interrupt routines must be allocated priorities that define the sequence in which the interrupt routines are executed:
PRIO=<value>

There are priorities from 1 to 128. Priority 1 corresponds to the highest priority.

Example:

```
Program code                                Comment
...
N20 SETINT(3) PRIO=2 ABHEBEN_Z          ; If input 3 switches, then interrupt routine
                                            "ABHEB_Z" should start.
N30 SETINT(2) PRIO=2 ABHEBEN_X          ; If input 2 switches, then interrupt routine
                                            "ABHEB_X" should start.
...
```

The interrupt routines are executed in the sequence of the priority values if the inputs become available simultaneously (are energized simultaneously): First "ABHEBEN_Z", then "ABHEBEN_X".

REPOS-query

With interrupt routines, sequences may be generated for which there is no unambiguous return to an interruption point in the block processing sequence (REPOS).

The system variable $P_REPINF can be used to scan the ASUB to determine whether a REPOS is possible.

Value	Meaning
0	Repositioning with REPOS not possible because: • Not called in the ASUB • ASUB ran from reset status • ASUB ran from JOG
1	Repositioning with REPOS possible in ASUB

Determining the cause of activation

With the system variable $AC_ASUP the cause leading to the activation of an interrupt routine is specified bit-coded and can be read in the part program and in synchronized actions (see "User-specific ASUB for RET and REPOS > Programming (Page 632)").

Flexible programming

The following commands help in the flexible programming of interrupt routines:

Command	Meaning
SAVE	If the SAVE command has been used to define the interrupt routine, the G codes, frames and transformations previously active in the interrupted part program become operative again as soon as the interrupt routine is ended.
DISABLE	The DISABLE command can be set to protect part program sections from being interrupted by an interrupt routine. The assignment interrupt signal ↔ part program is maintained but the interrupt routine no longer responds to the 0/1 signal transition.
ENABLE	The DISABLE command can be reset with the ENABLE command. Interrupt routines are not activated until the next 0/1 transition of the interrupt signal.
CLRINT	Clear assignment interrupt signal ↔ part program

References

Programming Manual, Job Planning; Section: "Flexible NC Programming" > "Interrupt routine (ASUB)"

8.10.4 Restrictions

Cross-mode Start of interrupt routines

Requirement

MD11602 $MN_ASUP_START_MASK, at least bit 0 = 1

The following data must also be taken into account:

- MD11600 $MN_BAG_MASK
- MD11604 $MN_ASUP_START_PRIO_LEVEL
- Interrupt assignment priority

Recommended settings

NC-specific machine data:

- MD11600 $MN_BAG_MASK = 'H11'
- MD11602 $MN_ASUP_START_MASK = 'H101'
- MD11604 $MN_ASUP_START_PRIO_LEVEL = 7

Channel-specific machine data for the channel in which the ASUB is started or generally for all channels:

- MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK = ’H3F’
- MD20115 $MC_IGNORE_REFP_LOCK_ASUP = ’HFFFFFFFF’

8.10.5 Examples

Activation of an interrupt routine via synchronous action

1. Define number of active digital inputs/outputs:

   MD10350 $MN_FASTIO_DIG_NUM_INPUTS=3

   MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS=3
2. Generate short-circuit with the following MD setting:

   MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT[0]='H0102B102'

   MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT[1]='H0202B202'
3. HW assignment of the external NC input byte for NC program interrupt:

   MD21210 $MC_SETINT_ASSIGN_FASTIN=2 ; better 1 byte more than needed
4. Define input as ASUB trigger:

   ```
   SETINT(1) PRIO=1 SYNCASUP
   ```
5. `IDS=1 EVERY $$AC_PATHN>=0.5 DO $A_OUT_[9]=1`

8.11 User-specific ASUB for RET and REPOS

8.11.1 Function

Function

The NCK software supplied contains preprogrammed processes (internal ASUBs) for implementation of the RET and REPOS functions. They can be replaced by user-specific ASUBs written by the machine tool manufacturer.

⚠ DANGER
The machine manufacturer is responsible for the contents of ASUB routines used to replace ASUP.SYF supplied by Siemens.

Installation

In the manufacturer directory _N_CMA_DIR or in the user directory _N_CUS_DIR **a** routine with the name "_N_ASUP_SPF" can be loaded. These must implement the actions desired by the user for the functions RET and REPOS.

8.11.2 Parameter assignment

Activating

The parameters for the activation of the user-specific routine "_N_ASUP_SPF" are set with the machine data:

MD11610 $MN_ASUP_EDITABLE (activation of a user-specific ASUP program).

Bit 0 and bit 1 specify, which of the internal system routines are to be replaced by the user-specific ASUB:

Binary value	Description
0	**Neither** in case RET **nor** in case of REPOS the user-specific routine _N_ASUP_SPF is activated.
1	The user-defined routine is activated for **RET**, the routine provided in the system is activated for REPOS.
2	The user-defined routine is activated for **REPOS**, the routine provided in the system is activated for RET.
3	**As** in case RET **and** also in case of REPOS the user-specific routine is activated.

Bit 2 defines in which directory the user-specific routine is to be searched first in case of activation.

Bit	Value	Description
2	0	The user-specific routine is searched first in the user directory **_N_CUS_DIR**.
	1	The user-specific routine is searched first in the manufacturer directory **_N_CMA_DIR**.

Defining a level of protection

If a user-specific ASUB is to be used for RET and/or REPOS, i.e. when:
MD11610 $MN_ASUP_EDITABLE ≠ 0
then a level of protection can be defined for the user-specific routine "_N_ASUP_SPF". The level of protection can have values in the range 0 - 7.

The setting is done via the following machine data:

MD11612 $MN_ASUP_EDIT_PROTECTION_LEVEL (level of protection of the user-specific ASUB)

For further information about protection levels, refer to:
References:
Startup guide; level of protection concept

Behavior when the single block processing is set

Via the following machine data it can be set, that despite a set single block processing the internal ASUB or the user-specific "_N_ASUP_SPF" is processed without interruption:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (Prevent single block stop)

Bit	Value	Description
0	0	A stop is done in each ASUB block.
	1	The ASUB is processed without interruption.

8.11.3 Programming

Determining the cause of the ASUB activation

The cause of the activation of the ASUB can be read bit-coded via the system variable $AC_ASUP.

Continuation

When using the system ASUB, the behavior for the continuation after execution of the actions is permanently specified within the ASUB:

- System ASUB 1 → continuation with `RET` (subprogram return)
- System ASUB 2 → continuation with `REPOS` (repositioning)

The description of the system variables specifies the behavior with regard to the system ASUB for each cause at "Continued for".

Note

Continued for user-specific ASUB

It is recommended for user-specific ASUBs that the appropriate continuation of the system ASUB be retained.

Cause: Change of operating mode ($AC_ASUP, bit 9 == 1)

At a change of operating mode, the continuation depends on the machine data:

MD20114 $MC_MODESWITCH_MASK (interruption of MDA through mode change)

- Bit 0 == 0: System ASUB 1 → continuation with `RET`
- Bit 0 == 1: System ASUB 2 → continuation with `REPOS`

References

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

List Manual, System Variables

8.12 Single block

Block-by-block processing

With the single-block function, the user can execute a part program block-by-block.

Single-block types

There are 3 types of setting for the single-block function:

- SBL1 := IPO single block
  When the SLB1 function is active, machining stops or pauses after each machine action block (Ipo block).
- SBL2 := Decode single block
  When the SLB2 function is active, machining always stops or pauses after each part program block. If a part program block is processed in several IPO blocks, machining stops after every Ipo block. Thread cutting is an exception.
- SBL3 := Decode single block
  As for SLB2, but machining also stops in the part program blocks of the cycles.

1. Stopping after every block is undesirable in many situations and/or with certain blocks.
   - 1. Example:
     Change after jog operation, if reorganization and/or repositioning
     is not possible, MD10702, bits 6 and 7.
     If a stop occurs in a block at the end of block, which cannot be
     reorganized and/or repositioned, in this situation
     Jog mode cannot be selected.
   - 2. Example:
     Change after JOG operation to a STOPRE block,
     MD10702, bits 6 and 7
     If AUTO mode is changed to Jog mode while a
     STOPRE block is active, in addition to system ASUB2, a continuation start will be followed by
     one residual block and one or possibly (with
     decoder single block) two more STOPRE blocks. A logic operation,
     which always triggers a part program start in single block
     and then always changes to Jog mode, remains
     at the STOPRE block indefinitely.
   - 3. Example:
     DISPOF: Deactivate block display, MD10702, bits 6 and 7
     If DISPOF is programmed in a subroutine, the block display
     is suppressed. The operator must continuously
     press Start blindly in the single block up to the end of the subroutine.

2. When single block is deactivated there is no stop at end of block.
3. When STOPRE blocks are displayed, the main run and preprocessing are synchronized in the decoding single block.

The following sections describe how to control the behavior of single blocks and prevent stops in particular situations.

8.12.1 Decoding single block SBL2 with implicit preprocessing stop

Asynchronicity

As a result of preprocessing of part program blocks, the reference between the current block display relative to the main run status of the NCK and the variable values displayed on the HMI can be lost. The operator display then shows implausible variable values.

Preprocessing stop for each block

A preprocessing stop is executed for active SBL2 with each block with the channel-specific setting data SD42200 $SC_SINGLEBLOCK2_STOPRE (activate debug mode for SBL2). This suppresses preprocessing of part program blocks and maintains the relationship between the current block display and the variable values display.

Note

This variant of SBL2 does not maintain an accurate contour. In other words, as a result of the preprocessing stop, a different contour may be generated from the one created without single-block mode or with SBL1.

Application: Debug mode for testing part programs.

8.12.2 Single block stop: Suppression using SBLOF

Single block off

Programs characterized by the SBLOF language command are executed completely in one block as with every type of single block.

SBLOF is also valid in the called subroutines.

SBLOF

Example for subroutine without stop in single block:

```
PROC EXAMPLE SBLOF
G1 X10
RET
```

At the return command, the decision is made whether to stop at the end of the subprogram:

Return jump with M17	Stop at the end of the subprogram
Return jump with RET	No stop at end of subroutine

SBLOF in the program

SBLOF alone must remain in the block. Single-block stop is deactivated from this block onwards up to the next programmed SBLON or up to the end of the active subroutine level.

If SBLOF is active, then this definition is also valid in the called subroutines. SBLON

Example for an area in single block mode

The area between N20 and N60 is executed as one step in single-block mode.

```
N10 G1 X100 F1000
N20 SBLOF                          ; Deactivate single block
N30 Y20
N40 M100
N50 R10=90
N60 SBLON                          ; Reactivate single block
N70 M110
N80 ...
```

Asynchronous subprograms

The asynchronous subroutines ASUP1.SYF and ASUP2.SYF started system-internally in REORG/REPOS can process the system ASUP in one step through the programming of SBLOF.

Example: ASUP.SPF:

```
N10 SBLOF
N20 IF $AC_ASUP==’H200’
N30 RET                            ; No REPOS on mode change
N40 ELSE
N50 REPOSA                         ; REPOS in all other cases
N60 ENDIF
N70 RET
```

Constraints

- The current block display can be suppressed in cycles with DISPLOF.
- If DISPLOF is programmed together with SBLOF, then the cycle call continues to be displayed on single-block stops within the cycle.
- The preset behavior of asynchronous subroutines in single block mode specified in MD20117 MC_IGNORE_SINGLEBLOCK_ASUP (process interrupt program fully despite single block) can be overwritten on a program-specific basis using SBLOF.

Cycle

Example 1: A cycle is to act like a command for a user.

Main program:

```
N10 G1 X10 G90 F200
N20 X-4 Y6
N30 CYCLE1
N40 G1 X0
N50 M30
```

Program cycle:1

```
N100 PROC CYCLE1 DISPLOF SBLOF        ; Suppress single block
N110 R10=3*SIN(R20)+5
N120 IF (R11 <= 0)
N130 SETAL(61000)
N140 ENDIF
N150 G1 G91 Z=R10 F=R11
N160 M17
```

CYCLE1 is processed for an active single block, i.e., the Start key must be pressed once to process CYCLE1.

Example 2: An ASUB, which is started by the PLC in order to activate a modified zero offset and tool offsets, is to be executed invisibly.

```
N100 PROC ZO SBLOF DISPLOF
N110 CASE $P_UIFRNUM OF 0 GOTOF
_G500 1 GOTOF _G54 2 GOTOF _G55 3 GOTOF _G56 4
GOTOF _G57 DEFAULT GOTOF END
N120 _G54: G54 D=$P_TOOL T=$P_TOOLNO
N130 RET
N140 _G54: G55 D=$P_TOOL T=$P_TOOLNO
N150 RET
N160 _G56: G56 D=$P_TOOL T=$P_TOOLNO
N170 RET
N180 _G57: G57 D=$P_TOOL T=$P_TOOLNO
N190 RET
N200 END: D=$P_TOOL T=$P_TOOLNO
N210 RET
```

8.12.3 Single block stop: inhibit according to situation

Suppress stopping in single cases

Depending on

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop)

setting bits 0 to 12 = 1 can suppress stopping at the end of the block during the following machining processes.

Program execution must not stop after single blocks in the case of the following even if non-modal processing is selected:

1. During an internal ASUB
2. During a user ASUB
3. Subprograms with the attribute DISPLOF
4. Intermediate blocks
5. Block search group blocks
6. Init blocks
7. Blocks that cannot be reorganized
8. Blocks that cannot be repositioned
9. A repositioning block which contains no traversing information
10. A preprocessing / main run / synchronization block due to REORG
11. At a tool selection block
12. At a GET block
13. During a single block type 2

Sequence

If an ASUB is activated during the single block, for example, execution of the ASUB is completed. The deceleration does not take place until after the end of the ASUB or the first IPO block in which single-block suppression is not activated. If the velocity is too large for the deceleration to be performed in this block (with continuous-path mode G64 active), `G64`), further block changes are allowed.

For decoding single block, MD10702 is only effective with "internal ASUB", "user ASUB" and "subprograms with the attribute DISPLOF". In these cases, it is already clear at the time of interpretation that the block belongs to one of the above categories. In these cases, further blocks can be generated.

SBLON in ASUB

The single block stop of an internal ASUP or user ASUP that is suppressed with MD10702 $MN_IGNORE_SINGLEBLOCK_MASK can be reactivated in ASUP by programming `SBLON`.

This functionality can be suppressed again with MD20117 $MC_IGNORE_SINGLEBLOCK_ASUB. The `SBLON` command then becomes ineffective.

Supplementary conditions

The following restriction applies to decoding single block SBL2:

- Block search approach blocks
- Block not in ASUB; DISPLOF, SBLOF
- Non-reorganizable or non-repositionable blocks
- Blocks that are not generated in the Interpreter, e.g. intermediate blocks

8.12.4 Single-block behavior in mode group with type A/B

Classifying channels

One mode group channel must be classified as a single-block control channel (KS), while the other mode group channels must be classified as dependent channels (KA) via interface signal. Type A or type B single-block behavior can be selected for KA channels.

Type A determines Stop (analogous to STOP key).

Type B determines Stop (analogous to stop at block limit).

Channel classification

In **one** channel (**KS**) in a mode group, the user should select single-block (NST DB21 ... DBX0.4 (activate single block)). Single-block type A or B refers to **other** channels (KA) of a mode group.

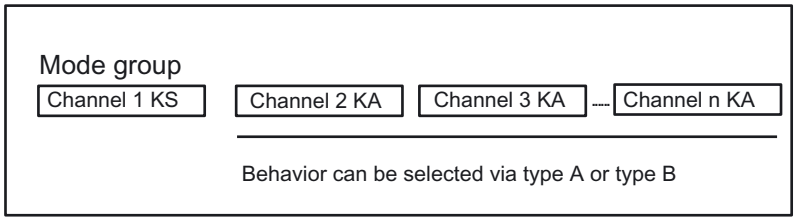


Figure 8-13 Channel classification for single block in mode group 1

Type A, IS DB11, … DBX1.7=1 (single-block type A)

- All channels are stopped
- All channels receive a start (Start key)
- Channel KS stops at the end of the block (due to single-block)
- Channels KA receive a **STOP** (analogous to Stop key)
- All channels are stopped (deceleration phase of all KAs)

Type B, IS DB11, ... DBX1.6=1 (single-block type B)

- All channels are stopped
- All channels receive a start
- Channel KS stops at the end of the block
- Channels KA receive a **STOPATEND**
  (analogous with NST DB21, ... DBX7.2 ("NC stop at the block limit)
- All channels are stopped at a block limit (at some point in time)

8.13 Program control

Options

1. Function selection (via operator interface or PLC)
2. Activation of skip levels
3. Adapting the size of the interpolation buffer
4. Program display modes via an additional basic block display
5. Execution from external source (buffer size and number)
6. Execution from external subroutines

8.13.1 Function selection (via operator panel front or PLC)

User interface or PLC

The user can control part program execution via the operator panel front or PLC.

Selection, activation, feedback

Selection
Different functions are available under the Program control soft key. Selection affects an interface signal in the PLC. These signals are to be understood as selection signals from the user interface, and do not activate the selected function.

Activation
These signal states must be transferred to another area of the data block to activate the selected function. With program control by the PLC the signals are to be set directly.

Feedback
The activated functions are partly signaled back to the PLC from the NCK (see also Section "Z1: NC/PLC interface signals (Page 1791)").

Table 8- 4 Program control: Interface signals

Function	Selection signal	Activation signal	Feedback signal
SKP skip block 0 to 7 SKP skip block 8 to 9	DB21, ... DBX26.0 - 26.7 DB21, ... DBX27.0 - 27.1	DB21, ... DBX2.0 - 2.7 DB21, ... DBX31.6 - 31.7	- - -
DRY dry run feedrate	DB21, ... DBX24.6	DB21, ... DBX0.6	DB21, ... DBX318.6
ROV Rapid traverse override	DB21, ... DBX25.3	DB21, ... DBX6.6	- - -
Single block:	Preselection of SBL1, SBL2 or SBL3 via program control display of HMI		
SBL1: Action single block	HMI operator panel	DB21, ... DBX0.4	- - -
SBL2: Decoding single block			
SBL3: In cycle			
M01 (Programmed stop)	DB21, ... DBX24.5	DB21, ... DBX0.5	DB21, ... DBX32.5
Associated M01	DB21, ... DBX24.4	DB21, ... DBX30.5	DB21, ... DBX318.5
DRF selection	DB21, ... DBX24.3	DB21, ... DBX0.3	DB21, ... DBX33.3
PRT program test	DB21, ... DBX25.7	DB21, ... DBX1.7	DB21, ... DBX33.7

References:
Operating Manual HMI Advanced "Operating Area, Machine"

8.13.2 Activation of skip levels

Function

It is possible to skip blocks which are not to be executed every time the program runs. Blocks to be skipped are indicated in the part program by the character "/" before the block number.

The skip levels in the part program are specified by "/0" to "/9".

Only one skip level can be specified for each part program block.

Parameter assignment

The number of skip levels is defined using machine data:

MD51029 $MM_MAX_SKP_LEVEL (max. number of skip levels in the NC program)

Programming

Blocks which are not to be executed in every program pass (e.g. program test blocks) can be skipped according to the following schematic.

```
Program code          Comment
/N005                 ; Block skipped, (DB21,... DBX2.0) 1st skip level
/0 N005               ; Block skipped, (DB21,... DBX2.0) 1st skip level
/1 N010               ; Block skipped, (DB21,... DBX2.1) 2nd skip level
/2 N020               ; Block skipped, (DB21,... DBX2.2) 3rd skip level
/3 N030               ; Block skipped, (DB21,... DBX2.3) 4th skip level
/4 N040               ; Block skipped, (DB21,... DBX2.4) 5th skip level
/5 N050               ; Block skipped, (DB21,... DBX2.5) 6th skip level
/6 N060               ; Block skipped, (DB21,... DBX2.6) 7th skip level
/7 N070               ; Block skipped, (DB21,... DBX2.7) 8th skip level
/8 N080               ; Block skipped, (DB21,... DBX31.6) 9th skip level
/9 N090               ; Block skipped, (DB21,... DBX31.7) 10th skip level
```

Activation

The 10 skip levels "/0" to "/9" are activated by the PLC setting the PLC → NCK interface signals.

The function is activated from the HMI via the "Program control" menu in the "Machine" operating area:

- For skip levels "/0" to "/7":

  Via the interface HMI → PLC DB21, ... DBB26 (skip block selected).

- For skip levels "/8" to "/9":

  Via the interface HMI → PLC DB21, ... DBX27.0 to DBX27.1.

References:
Operating Manual

Note

The levels to be skipped can only be changed when the control is in the STOP/RESET state.

8.13.3 Adapting the size of the interpolation buffer

MD28060

The channelspecific interpolator executes prepared blocks from the interpolation buffer during the part program run. The maximum number of blocks requiring space in the interpolation buffer at any given point in time is defined by the memory configuring MD28060 $MM_IPO_BUFFER_SIZE (number of NC blocks in the IPO buffer (DRAM)). For some applications it may be meaningful not to use the full buffer capacity in order to minimize the "interval" between preparation and interpolation.

SD42990

The number of blocks in the interpolation buffer can be restricted dynamically to a **smaller** value than in MD28060 $MC_MM_IPO_BUFFER_SIZE (number of NC blocks in the IPO buffer (DRAM)), minimum 2 blocks, with the setting data SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER (max. number of blocks in the IPO buffer).

Values of setting data SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER:

Value	Effect
< 0	No interpolation buffer limit active. The max. possible interpolation buffer as set in MD 28060: MM_IPO_BUFFER_SIZE is activated.
or 1	The minimum permissible interpolation buffer with 2 blocks is activated.
< < MM_IPO_BUFFER_SIZE	The interpolation buffer is activated with no more than the maximum specified number of blocks.
>= MM_IPO_BUFFER_SIZE	The interpolation buffer is activated with the number of blocks specified in MD 28060: MM_IPO_BUFFER_SIZE.

Note

If SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER is set in the part program, the interpolation buffer limitation takes effect immediately if the block with the SD is being preprocessed by the interpreter.

This means that the limitation of the IPO buffer may take effect a few blocks before the intended limitation (see also MD 28070 $MC_MM_NUM_BLOCKS_IN_PREP).

To avoid premature activation and to make the limitation of the IPO buffer take effect in synchronism with the block, a STOPRE (preprocessing stop) must be programmed before the SD is set in the part program.

Validity

SD42990 $SC_MAX_BLOCK_IN_IPOBUFFER has global, channel-specific validity and can also be modified in a part program. This modified value is maintained at program end. If this setting data is to be reset again on defined events, a so-called event-driven program must be created to do this. For example, this setting data could always be set to a predefined value on RESET.

Application

The IPO buffer limitation can be used whenever the number of blocks between block preparation and interpolation must be minimized, e.g., when actual positions in the part program must be read and processed for other purposes.

Example

```
N10 ...
N20 ...
..........
N100 $SC_MAX_BLOCKS_IN_IPOBUFFER = 5           ;  Limitation of IPO buffer to 5 NC
                                                  blocks
N110 ...
N120 ...
............
N200 $SC_MAX_BLOCKS_IN_IPOBUFFER = -1          ;  Cancellation of the IPO buffer
                                                  limitation
N210 ...
............
```

8.13.4 Program display modes via an additional basic block display

Basic block display (only for ShopMill/ShopTurn)

A second so-called basic block display can be used with the existing block display to show all blocks that produce an **action on the machine**.

LookAhead basic block display

The actually approached end positions are shown as an absolute position. The position values refer either to the workpiece coordinate system (WCS) or the settable zero system (SZS).

The number of LookAhead display blocks stored in the display buffer depends on the number of prepared blocks in the NCK preprocessing buffer in the relevant processing state. If a preprocessing stop is processed, the number of display blocks is reduced to zero and increases again after the stop is acknowledged. In the case of REORG events (e.g. mode change, ASUB start), the display blocks stored for LookAhead are deleted and preprocessed again afterwards.

Processed values

Values processed in the basic block display coincide with the:

- Selected tools
- Feedrate and spindle speed
- Actually approached position values
  Exceptions:
  With active tool radius compensation, deviations can occur.
  For modulo axes, the programmed value is displayed in the basic block display. This value can also lie outside the modulo range.

Note

Generally the positions are represented in the WCS or the SZS.

The basic block display can be activated or deactivated with setting data

SD42750 $SC_ABSBLOCK_ENABLE.

8.13.5 Basic block display for ShopMill/ShopTurn

Configure basic block display

The basic block display can be configured via the following machine data:

NCK machine data for basic block display	Significance:
MD28400 $MC_MM_ABSBLOCK	Activate basic block display
MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[2]	Size of display buffer
Display machine data	Position values to be set:
MD9004 $MM_DISPLAY_RESOLUTION	For metric measurements
MD9011 $MM_DISPLAY_RESOLUTION_INCH	For inch measurements
MD9010 $MM_SPIND_DISPLAY_RESOLUTION	Settable coordinate system for spindle display resolution
MD9424 $MM_MA_COORDINATE_SYSTEM	For actual value display in WCS or SZS

These display machine data are copied to NCK machine data MD17200 $MN_GMMC_INFO_UNIT[0] to MD17200 $MN_GMMC_INFO_UNIT[3]. allowing them to be accessed from the NCK.

Activating

The basic block display is activated by MD 28400 $MC_MM_ABSBLOCK by means of Power On. If MD28400 $MC_MM_ABSBLOCK is set to 1, a channelspecific display buffer (FIFO) is created during power-up.

Size of display buffer (FIFO) = (MD28060 $MC_MM_IPO_BUFFER_SIZE + MD28070 $MC_MM_NUM_BLOCKS_IN_PREP) multiplied by 128 bytes. This corresponds to a size of 6KB in the machine data default setting.

Optimize size of display buffer:
The memory requirement can be optimized by entering a value between 128 and 512. The display blocks preprocessed in the display buffer are transferred to the HMI via a configurable upload buffer.

Maximum size of upload buffer is obtained by multiplying (MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[0] + MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[1] + 1) by the block length configured in MD28400 $MC_MM_ABSBLOCK.

The number of blocks **before** the current block is configured in MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[0] and the number of blocks **after** the current block is configured in MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[1].

Constraints

If the length of a display block configured in MD28400 $MC_MM_ABSBLOCK is exceeded, this display block is truncated accordingly. This is represented by string "..." at the end of the block.

For preprocessed cycles (MD10700 $MN_PREPROCESSING_LEVEL > 1), the display block contains **only** axis positions.

Additional boundary conditions for the basic block display:

- Modal synchronized action blocks with absolute values are not taken into account.
- The basic block display is deactivated during block search with or without computation.
- Polar coordinate programming is not shown in Cartesian system.

Radius / diameter values

Diameter values shown in the basic block display and position display may be needed as a radius for internal calculation. These values for measurements in radius/diameter according to G code group 29 can be manipulated using the following options:

- G code DIAMCYCOF (expansion of channel-specific diameter programming)
  This G code deactivates the channel-specific diameter programming during the cycle execution. In this way, computations in the cycle can always be done in the radius. The position display and the basic block display are continued according to the state of the diameter programming before DIAMCYCOF.
  In the basic block display, the last displayed value is retained.
- G code DIACYCOFA[AX] (axis-specific diameter programming)
  This G code deactivates the axis-specific diameter programming during the cycle execution. In this way, computations in the cycle can always be done in the radius. In the position display and in the basic block display, this continues according to the state of the diameter programming before DIACYCOFA[AX].
  In the basic block display, the last displayed value is retained.
- MD27100 $MC_ABSBLOCK_FUNCTION_MASK

Bit0 = 1	Transverse axis setpoints are always shown as diameter values in the basic block display.

Behavior while the compressor is active

With active compressor and G/Code group 30 not equal to COMPOF, two display blocks are generated. The

- first contains the G/Code of the active compressor.
- The second contains the string "..." as character for missing display blocks.

Example:

```
G0 X10 Y10 Z10          ;  Block to be preprocessed for the basic block display
COMPCAD                 ;  Compressor for optimized surface quality (CAD prog.) A
...                     ;  string as character for missing display blocks
```

To avoid bottlenecks in the NCK performance, the basic block display is deactivated automatically. As a sign that the display blocks are missing, a display block with the string "..." is generated.
All display blocks are always generated in the single block.

8.13.6 Structure for a DIN block

Structure of display block for a DIN block

Basic structure of display block for a DIN block

- Block number/label
- G function of the first G group
  (only if changed as compared to the last machine function block).
- Axis position
  (sequence corresponding to MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in the channel)).
- Further modal G functions
  (only if changed as compared to the last machine function block).
- Other addresses as programmed.

The display block for the basic block display is directly derived from the programmed part program blocks according to the following rules:

- Macros are expanded.
- Skip identifiers and comments are omitted.
- Block number and labels are transferred from the original block, but omitted if DISPLOF is active.
- The number of decimal places is defined in display machine data MD 9004, MD 9010 and MD 9011 via the HMI.

HMI display machine data	Access in NCK machine data
MD9004 $MM_DISPLAY_RESOLUTION	MD17200 $MN_GMMC_INFO_NO_UNIT[0]
MD9011 $MM_DISPLAY_RESOLUTION_INCH	MD17200 $MN_GMMC_INFO_NO_UNIT[1]
MD9010 $MM_SPIND_DISPLAY_RESOLUTION	MD17200 $MN_GMMC_INFO_NO_UNIT[2]
MD9424 $MM_MA_COORDINATE_SYSTEM	MD17200 $MN_GMMC_INFO_NO_UNIT[3]

- Programmed axis positions are represented as absolute positions in the coordinate system (WCS / ENS) specified in MD9424 $MM_MA_COORDINATE_SYSTEM (coordinate system for actual value display)

Note

The modulo correction is omitted for modulo axes, which means that positions outside the modulo range can be displayed. It also means that the basic block display differs from the position display in which values are always moduloconverted.

Examples

Comparisons between display block (original block) and basic block display:

- **Programmed positions** are displayed as absolute.
  The addresses AP/RP are displayed with their programmed values.

Original block:	Display block:
N10 G90 X10.123	N10 X10.123
N20 G91 X1	N20 X11.123

- **Address assignments** (non-DIN addresses) are displayed in the form <address> = <constant>.

Original block:	Display block:
N110 R1 = -67.5 R2 = 7.5	
N130 Z = R1 RND = R2	N130 Z-67.5 RND = 7.5

- **Address indices** (address extensions) are displayed as constants <address> [<constant>] = <constant>.

Original block:	Display block:
N220 DEF AXIS AXIS_VAR = X	
N240 FA[AXIS_VAR] = R2	N240 FA[X] = 1000

- **DIN addresses without address extension** are displayed in the form <din_address> <constant>.

Original block:	Display block:
N410 DEF REAL FEED = 1.5	
N420 F = FEED	N420 F1.5

The following applies for **H functions**: Each programmed value is display irrespective of the output type to the PLC.
(MD22110 $MC_AUXFU_H_TYPE_INT (type of H auxiliary function is integer)).

- For **Tool selection by tool command**
  Display information is generated in the form T<value> or T=<string>. If an address extension has been programmed, this is displayed as well.

  If several spindles have been configured or the "Tool change via master toolholder" function (MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER (toolholder number)) is active, the T number is always output with address extension.

  If no address extension has been programmed, the number of the master spindle or the master toolholder is used instead (T<spindle_number/tool_holder>=).
- For the **Spindle programming** via S, M3, M4, M5, M19, M40 - M45 and M70 (or MD 20094 $MC_SPIND_RIGID_TAPPING_M_NR (M function for switching over in the controlled axis operation)) the following regulation applies regarding the address extension:
  If an address extension has been programmed, then this is also resolved.

  If several spindles have been configured, then the address extension is also output.
  If no address extension has been programmed, the number of the master spindle is used (S<spindle_number>=).
- **Indirect G code programming** in form G[<group>] = <printout> is substituted by the corresponding G code.

Original block:	Display block:
N510 R1=2	
N520 G[8]= R1	N520 G54

- **Modal G codes** that do not generate an executable block are collected and output with the display block of the next executable block if permitted by the syntax (DIN block). If this is not the case (e.g. predefined subprogram call TRANSMIT), a separate display block containing the modified G codes is placed in front of the next executable block.

Original block:	Display block:
N610 G64	G64
N620 TRANSMIT	N620 TRANSMIT

- A display block is always generated for **part program lines** in which the addresses **F and FA** appear (including for MD22240 $MC_AUXFU_F_SYNC_TYPE = 3 (output time of the F functions)).

Original block:	Display block:
N630 F1000	N630 F1000
N640 X100	N640 X100

- The **display blocks generated for the block display** are derived **directly** from the programmed part program blocks. If intermediate blocks (e.g. tool radius compensation G41/G42, radius/chamfer RNDM, RND, CHF, CHR) are generated in the course of contour preprocessing, these are assigned the display information from the part program block on which the motion is based.

Original block:	Display block:
N710 Y157.5 G42	N710 Y157.5 G42
N720 Z-67.5 RND=7.5	N720 Z-67.5 RND=7.5

- With the **EXECTAB command** (processing a table of contour elements), the block generated by EXECTAB is shown in the display block.

Original block:	Display block:
N810 EXECTAB (KTAB[5])	N810 G01 X46.147 Z-25.38

- With the **EXECSTRING command**, the block generated via EXECSTRING is displayed in the display block.

Original block:
N910 DEF STRING[40] PROGSTRING = "N905 M3 S1000 G94 Z100 F1000 G55"
N920 EXECSTRING(PROGSTRING)

Original block:
N905 Z100 G55 G94 M3 S1000 F1000

8.13.7 Execution from external

Function

The "Execution from external" function can be used to execute programs that cannot be saved directly in the NC memory due to memory shortage from an external program memory.

Note

Protected cycles (_CPF files) cannot be processed with this function.

External program memory

External program memory can be found on the following data carriers:

- Local drive
- Network drive
- USB drive

Note

Execution from external source via USB interface

If external programs are to be transferred from an external USB drive via a USB interface, only the interface via X203 (named "TCU_1") can be used.

A USB FlashDrive cannot be recommended as a persistent storage medium.

Applications

- **Direct execution from external programs**

  In principle, any program that is accessible via the directory structure of the interface in the "Execution from external" HMI mode can be selected and executed.

- **Execution of external sub-programs from the part program**

  The external subroutine is called through the part program command `EXTCALL` with specification of a call path (optional) and the subroutine identifier (→ see "Execution from external subroutines (Page 654)").

Parameter assignment

A reloading memory (FIFO buffer) must be reserved in the dynamic NC memory for executing a program in the "Executing from external" mode (main program or subroutine).

Size of FIFO buffer

The size of the FIFO buffer is set in the machine data:

MD18360 $MN_MM_EXT_PROG_BUFFER_SIZE (FIFO buffer size for processing from external)

Default: 30 kbyte

Number of the FIFO buffer

One FIFO buffer must be provided each for all programs (main run or subroutine) that are executed simultaneously in the "Execution from external source" mode.

The number of the FIFO buffer is set in the machine data:

MD18362 $MN_MM_EXT_PROG_NUM (number of externally executed program levels executable simultaneously)

Behavior during RESET, POWER ON

External program calls are aborted through RESET and POWER ON and the concerned FIFO buffers are erased.

A program selected for "Execution from external source" remains selected for "Execution from external source" even after RESET / part program end. A POWER ON deletes the selection.

8.13.8 Execution from external subroutines

Function

Individual machining steps for producing complex workpieces may involve program sequences that require so much memory they cannot be stored in the NC memory.

In such cases, the user has the option of executing the program sequences as subroutines from an external program memory in the "Execution from external source" mode with the help of the `EXTCALL` part program instruction.

Preconditions

The following preconditions are applicable to the execution from external subroutines:

- The subroutines must be accessible via the directory structure of the operator interface.
- A reloading memory (FIFO buffer) must be reserved for each subprogram in the dynamic NC memory.

Parameter assignment

The path for the external subprogram directory can be preset using setting data:

SD42700 $SC_EXT_PROG_PATH (Program path for the EXTCALL external subprogram call)

The entire path of the program to be called along with the subprogram path or identifier specified during programming is derived therefrom.

Programming

An external subroutine is called by means of parts program command EXTCALL.

Syntax: `EXTCALL("<path/><program name>")`

Parameter:

`<path>`: Absolute or relative path data (**optional**)
Type: STRING

`<program name>`: The program name is specified without prefix "_N_".
The file extension ("MPF", "SPF") can be attached to program names using the "_" or "." character (**optional**).
Type: STRING

Note

Path specification: Short designations

The following short designations can be used to specify the path:

- **LOCAL_DRIVE:** for local drive
- **CF_CARD:** for CompactFlash Card
- **USB:** for USB front connection

CF_CARD: and **LOCAL_DRIVE:** can be alternatively used.

EXTCALL call with absolute path name

If the subprogram exists at the specified path, it will be executed following the EXTCALL call. If it does not exist, program execution is cancelled.

EXTCALL call with relative path name/without path name

In the event of an EXTCALL call with a relative path name or without a path name, the available program memories are searched as follows:

- If a path name is preset in SD42700 $SC_EXT_PROG_PATH, the data specified in the EXTCALL call (program name or with relative path name) is searched for first, starting from this path. The absolute path results from linking the following characters:
  - The path name preset in SD42700
  - The "/" character as a separator
  - The subprogram path or identifier programmed in EXTCALL
- If the called subprogram is not found at the preset path, the data specified in the EXTCALL call is then searched for in the user-memory directories.
- The search ends when the subprogram is found for the first time. If the search does not produce any hits, the program is canceled.

Example

Execute from local drive

Main program:

Program code

```
N010 PROC MAIN
N020 ...
N030 EXTCALL ("ROUGHING")
N040 ...
N050 M30
```

External subprogram:

Program code

```
N010 PROC ROUGHING
N020 G1 F1000
N030 X= ... Y= ... Z= ...
N040 ...
...
...
N999999 M17
```

The "MAIN.MPF" main program is stored in NC memory and is selected for execution.

The "SCHRUPPEN.SPF" or "SCHRUPPEN.MPF" subprogram to subsequently loaded is on the local drive in the directory "/user/sinumerik/data/prog/WKS.DIR/WST1.WPD".

The subprogram path is preset in SD42700:

SD42700 $SC_EXT_PROG_PATH = "LOCAL_DRIVE:WKS.DIR/WST1.WPD"

Note

Without the path being specified in the SD42700, the `EXTCALL` operation for this example would have to be programmed as follows:

```
EXTCALL("LOCAL_DRIVE:WKS.DIR/WST1.WPD/SCHRUPPEN")
```

8.14 System settings for power-up, RESET / part program end and part program start

Concept

The behavior of the control can be set via the machine data for the following events:

- Run-up (Power On)
- Reset / part program end
- Part program start

The control-system response after:	Can be set with:
Run-up (POWER ON) *)	MD20110 $MC_RESET_MODE_MASK MD20144 $MC_TRAFO_MODE_MASK MD20150 $MC_GCODE_RESET_VALUES
RESET / part program end	MD20110 $MC_RESET_MODE_MASK MD20150 $MC_GCODE_RESET_VALUES MD20152 $MC_GCODE_RESET_MODE
Part program start	MD20112 $MC_START_MODE_MASK MD20110 $MC_RESET_MODE_MASK
*) see also POWER ON (Page 858)	

System settings after run-up

MD20110 $MC_RESET_MODE_MASK, bit 0 = 0 or 1

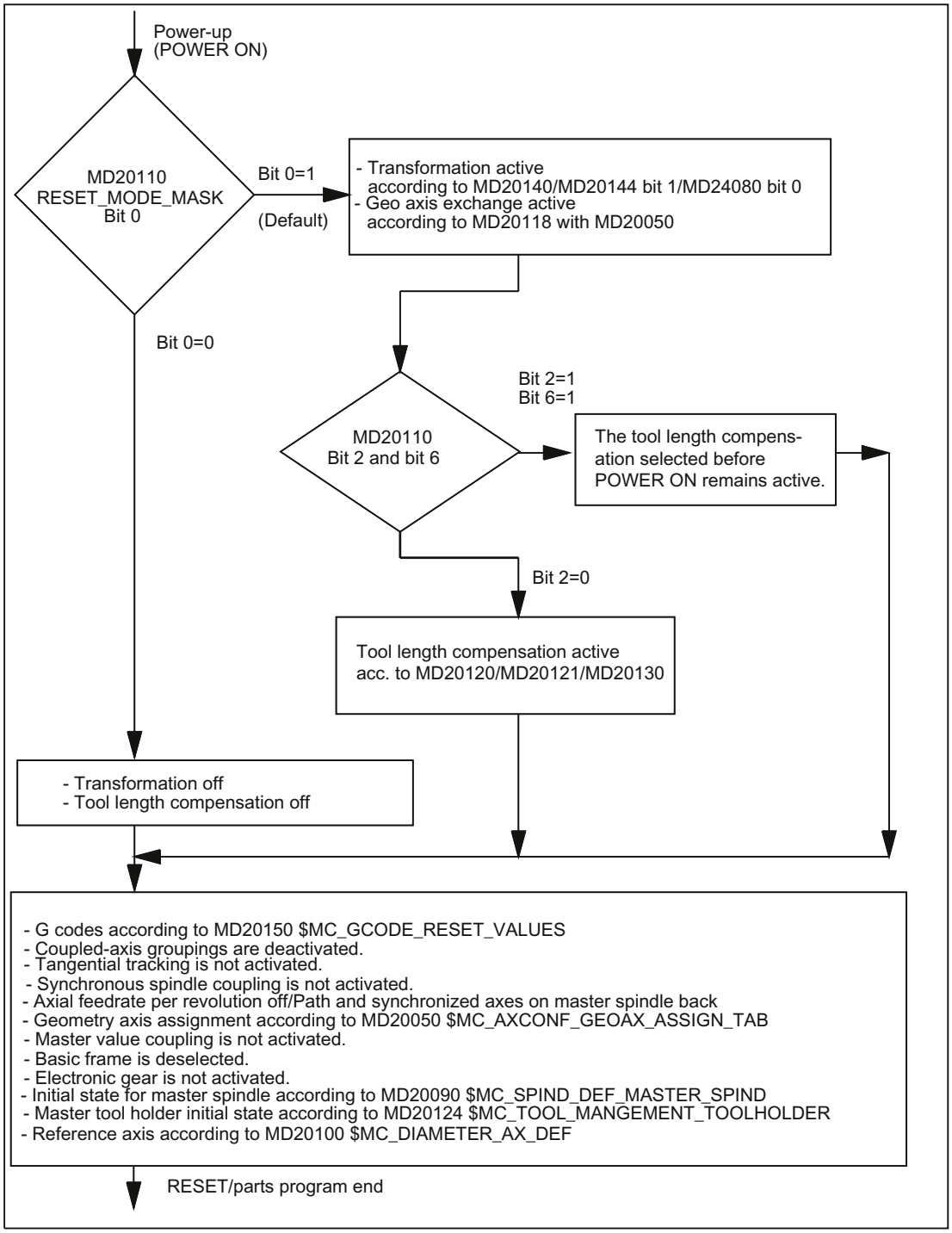


Figure 8-14 System settings after run-up

System settings after reset / part program end and part program start

MD20110 $MC_RESET_MODE_MASK, bit 0 = 0 or 1

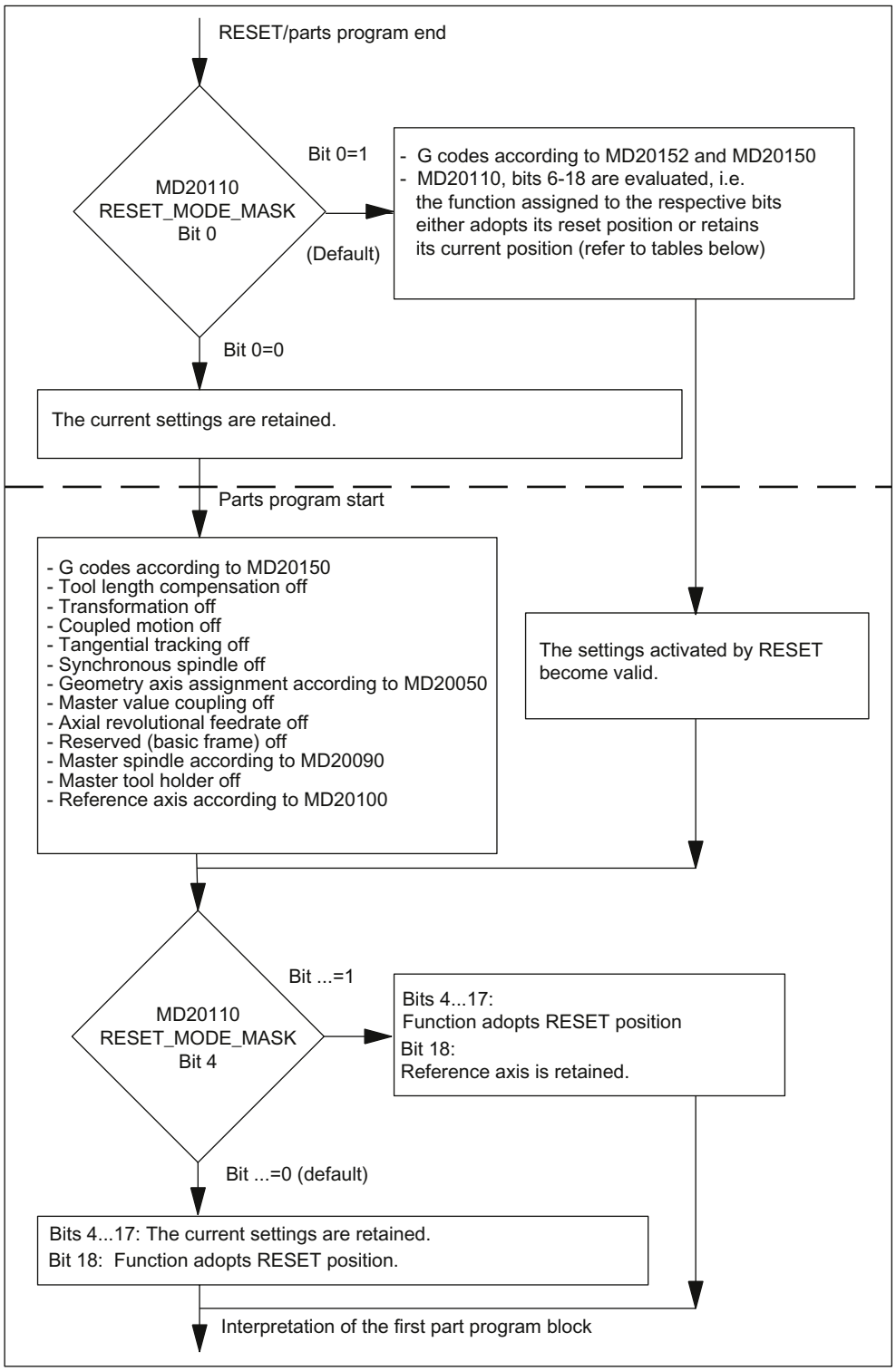


Figure 8-15 System settings after reset / part program end and part program start

G code effective after run-up and reset / part program end

The setting of the G code effective in every G group after run-up (power on) and reset / part program end is performed via the following machine data:

MD20150 $MC_GCODE_RESET_VALUES[<G group>] = <default-G code>

MD20152 $MC_GCODE_RESET_MODE[<G group>] = <value>

Value	Description: Per G group
0	The default-G code from MD20150 $MC_GCODE_RESET_VALUES takes effect.
1	The last active/current G code takes effect.

Control basic setting after run-up, reset / part program end and part program start

The definition of control initial setting after run-up, reset / part program end and part program start is performed via the following machine data:

- MD20110 $MC_RESET_MODE_MASK (definition of the control initial setting after **run-up** and **reset / part program end**)
- MD20112 $MC_START_MODE_MASK (definition of the control initial setting after **part program start**)

References

Detailed Machine Data Description

Relevant machine data

Machine data	Meaning
MD20120 $MC_TOOL_RESET_VALUE	Tool length compensation during run-up, reset / part program end
MD20121 $MC_TOOL_PRESEL_RESET_VALUE	Preselect tool on Reset
MD20130 $MC_CUTTING_EDGE_RESET_VALUE	Tool cutting-edge length compensation on run-up
MD20140 $MC_TRAFO_RESET_VALUE	Run-up transformation data block
MD20144 $MC_TRAFO_MODE_MASK	Selection of the kinematic transformation function
MD20150 $MC_GCODE_RESET_VALUES	Initial setting of the G groups
MD20152 $MC_GCODE_RESET_MODE	Reset behavior of the G groups
MD21330 $MC_COUPLE_RESET_MODE_1	Coupling cancellation response
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB	Assignment of geometry axis to channel axis
MD20118 $MC_GEOAX_CHANGE_RESET	Allow automatic geometry axis change

Example

Activate reset setting on reset:

- MD20110, bit 0 = 1
- MD20112 = 0

Transformation remains with reset / part program start:

- MD20110, bit 0 = 1
- MD20110, bit 7 = 1
- MD20112 = 0

Tool length compensation is retained after reset / part program start:

- MD20110, bit 4 = 1
- MD20110, bit 6 = 1
- MD20112 = 0

Active level (bit 4) and settable frame (bit 5) remain active after reset and are reset on part program start:

- MD20110, bit 4 = 1
- MD20110, bit 5 = 1
- MD20112, bit 4 = 1
- MD20112, bit 5 = 1

Note

MD20110/MD20112, bit 5 and bit 6

If MD20110/MD20112 are parameterized so that tool length compensation or a frame is active on a part program start in the automatic or MDI mode, the first programming of the axes must use absolute measurements (because of the traversing of the offset).

Exception: With MD42442/MD42440 the offsetting process for `G91` is suppressed.

8.14.1 Tool withdrawal after POWER ON with orientation transformation

Function

If a part program with a machining operation with tool orientation is aborted due to a power failure or reset, it is possible to select the previously active transformation and generate a frame in the direction of the tool axis after the control has run up (power on). The tool can then be retracted in JOG mode by means of a retraction movement towards the tool axis.

Requirement

The active measuring systems must have a machine reference for all the machine axes involved in the transformation. See Section "Position restoration at POWER ON (Page 1373)".

Parameter assignment

The following machine data must be set so that the last active transformation is retained after POWER ON:

- MD20144 $MC_TRAFO_MODE_MASK, bit 1 = 1
- MD20110 $MC_RESET_MODE_MASK, bit 0 = 1
- MD20110 $MC_RESET_MODE_MASK, bit 7 = 1

See also Section "System settings for power-up, RESET / part program end and part program start (Page 658)".

Programming

Wait for machine reference WAITENC

With the command WAITENC, the system waits channel-specifically in a program until there is valid machine reference for all the active measuring systems of the parameterized axes. See the "Requirement" section above. The parameter assignment of the axes is performed via:

MD34800 $MA_WAIT_ENC_VALID = 1

Application

In the user program (…/_N_CMA_DIR/_N_PROG_EVENT_SPF) to be called event-controlled when running up (requirement: MD20108 bit 3 = 1), the system must wait using the command `WAITENC` until the valid axis positions are available.

A frame that positions the tool axis in the direction of the X, Y or Z axis can then be generated using the NC language command `TOROTX`/`TOROTY`/`TOROTZ`.

Example

Orientation transformation and orientation axes with incremental encoders.

Configuration:	Meaning:
MD10720 $MN_OPERATING_MODE_DEFAULT [0] = 6	Run-up in JOG mode.
MD30240 $MA_ENC_TYPE [0, <axis>] = 1	Incremental measuring system.
MD34210 $MA_ENC_REFP_STATE [0, <axis>] = 3	Enable the restoration of axis positions for incremental encoders.
MD20108 $MC_PROG_EVENT_MASK = 'H9'	Activate event-controlled using program PROG_EVENT during run-up and at the start of the part program.
MD20152 $MC_GCODE_RESET_MODE [52] = 1	Obtain TOFRAME via reset.
MD20110 $MC_RESET_MODE_MASK = 'HC1'	Obtain transformation and tool offset via reset.
MD20144 $MC_TRAFO_MODE_MASK = 'H02'	Obtain transformation via POWER OFF.

Event-driven user program (…/_N_CMA_DIR/_N_PROG_EVENT_SPF):

```
Program code                              Comment
; Example with activation of the frame, which aligns the WCS in the tool direction, when running up
  and resetting with part program start.
IF $P_PROG_EVENT == 4                    ; Run-up
    IF $P_TRAFO <> 0                     ; Transformation has been selected.
        WAITENC                          ; Wait for valid axis positions of the orientation axes.
        TOROTZ                           ; Rotate the Z axis of the WCS towards the tool axis.
    ENDIF
    M17
ENDIF

IF $P_PROG_EVENT == 1                    ; Start of the part program.
    TOROTOF                              ; Reset the tool frame.
    RET
ENDIF
```

The WAITENC command essentially corresponds to the following program sequence (example for 5-axis machine with AB kinematics):

```
Program code                                        Comment
WHILE TRUE                                          ; Wait for a measuring system.
    IF (($AA_ENC_ACTIVE[X]==TRUE) AND ($AA_ENC_ACTIVE[Y]==TRUE) AND ($AA_ENC_ACTIVE[Z]==TRUE) AND
    ($AA_ENC_ACTIVE[A]==TRUE) AND ($AA_ENC_ACTIVE[B]==TRUE)) GOTOF GET_LABEL
    ENDIF
    G4 F0.5                                         ; 0.5 s wait time
ENDWHILE
:Position synchronization
GET_LABEL: GET(X,Y,Z,A,B,)
```

Continuing machining

AUTOMATIC mode

For automatic execution of programs in the AUTOMATIC mode, all the machine axes, whose actual position of the active measuring system has been restored, must be referenced.

MDI mode and overstore

In the MDI mode and for the overstore function, machining can also be performed, without referencing the axes, with restored positions. To do this, NC start with restored positions must be enabled explicitly for a specific channel:

MD20700 $MC_REFP_NC_START_LOCK = 2

Supplementary condition

Axes with incremental encoders and without actual value buffering

It is to be assumed that axes with incremental encoders and **without** actual value buffering are clamped with sufficient speed in the event of a power failure to prevent them drifting from their last position setpoint.

8.14.2 Tool retraction in the JOG retract submode

Function

The function "Retraction motion in the tool direction in the submode JOG retract", called "JOG retract" in the following, supports the manual tool retraction after a program abort through a power failure or a channel reset in the AUTOMATIC or MDA mode

Data to be restored

The following data that was active in the channel prior to the program abort must be restored for the retraction motion in the tool direction:

- Active tool offset
- Active machining plane
- Active tool carrier
- Active transformation data block with transformation parameters
- Data of the thread group with G33 or G331/G332
- Positions of the axes that are involved in the transformation

If this data is available after a program abort, it takes effect in the channel when the JOG retract submode is selected. In this case, the tool coordinate system (WCS) is aligned automatically in such a way that one of the geometry axes is in the direction of the tool axis. The tool retraction can then be performed through the manual traversing of this geometry axis.

Note

With JOG retract, only references to the above-mentioned data are backed up, not the data itself (e.g. tool offset values). If the data is changed before selecting JOG retract, the function is executed on the basis of the changed data.

Applications

Retraction of the tool after a program abort during:

- Machining with tool orientation with swivel cycle or orientation transformation
- Tapping with G33 (with compensating chuck and speed-controlled spindle with encoder)
- Interpolatory tapping with G331/G332 (without compensating chuck and position-controlled spindle)

Event-driven program calls

The following event-driven program calls are not executed in conjunction with JOG retract:

MD20108 $MC_PROG_EVENT_MASK	
Bit	**Event**
2	Operator panel reset
3	Power-up
5	Safety program during power-up

Programs started via interrupt signals of the NC are also not executed.

Note

Actual positions of the machine axes of transformations

If a transformation is active for JOG retract, a machine reference must be available for all machine axes involved in the transformation. See Section "Tool withdrawal after POWER ON with orientation transformation (Page 663)".

NC/PLC interface signal

When the relevant data is available for JOG retract so that the function can be executed, the following channel-specific NC/PLC interface signal is set:

The interface signal is reset with the start of the program execution in the AUTO or MDI mode.

Selection

The selection of the JOG retract submode is only possible when valid retract data is available. The display is realized via the channel-specific NC/PLC interface signal:

DB21, … DBX377.5 == 1 (retract data available)

Automatic changeover to the JOG retract submode

After power-up, normally the parameterized default mode is active:

MD10720 $MN_OPERATING_MODE_DEFAULT = <mode>

If retract data is available after power-up, the JOG retract submode can be activated automatically:

MD10721 $MN_OPERATING_MODE_EXTENDED = 1

"JOG retract active" feedback signal

The feedback signal that JOG retract is active, is performed via the NC/PLC interface signal:

DB21, … DBX377.4 == 1 (JOG retract active)

Selection via PLC user program

JOG retract can be selected from the PLC user program:

- Requirements:
  - DB21, … DBX377.5 == 1 (retract data available)
  - Channel status: Reset
- Selection: PI service "RETRAC" (see Section "P3: Basic PLC program for SINUMERIK 840D sl" > "Block descriptions" > "PI services" > "PI service: RETRAC (Page 1035)")

Tool retraction

The tool is retracted by manually traversing the geometry axis specified by the selection of JOG retract in the WCS. The manual traversing can also be performed incrementally. The JOG motion can then be stopped with NC stop and continued with NC start.

A spindle is also involved in the geometry axis group with G33 or G331/G332. The retraction motions is executed when the retraction axis or the spindle is traversed manually, i.e. when one axis of the group is traversed, all the axes involved are traversed. All the axes and spindles not involved in the geometry axis group can be manually traversed as required.

Traversing in the direction of the tool axis can be performed via the traversing keys as well as via handwheel.

NOTICE
The axis positions are synchronized or restored after a power failure. Once the tool has been retracted in the JOG retract submode, axes whose positions have been restored must be referenced (see also MD20700 $MC_REFP_NC_START_LOCK and MD34110 $MA_REFP_CYCLE_NR).

Tool retraction for thread cutting (G33) or tapping (G331, G332)

With tapping (G331, G332), the travel requests for the involved axes are summed up, i.e. as long as a travel request (traversing key) is present for an involved axis/spindle, travel is back to the contour. If a handwheel is selected for an involved axis/spindle and this is moved, then this is ignored as long as traversing via traversing key is active for another involved axis/spindle. If handwheels are selected for several axes or the spindle and these are moved, then the pulses of the handwheels are evaluated in the following order:

1. Retraction axis
2. Spindle
3. Other axis

The pulses of the other handwheels are ignored and only evaluated at standstill of the preceding handwheels.

With G33, G331, G332, the following spindle/axis requests are disabled during a possible retraction:

- Spindle start via DB31, ... DBX30
- FC18 for the spindle
- FC18 for an axis involved in the retraction
- Switching of the spindle or an axis involved in the retraction to the PLC-controlled axis
- Exchange of the spindle or an axis involved in the retraction to another channel or to the main run axis (command axis, oscillating axis, FC18 / concurrent axis)

MD10735 $MN_JOG_MODE_MASK, bit 8 = 0 can be used to set that the retraction axis can only be traversed in the positive direction during a retraction motion for G33, G331, G332. With bit 8 = 1, traversing can also be in the negative direction, however only to the starting point of the contour formed for the retract motion.

Deselection of JOG retract

The JOG retract submode is deselected with an operator panel reset or the channel-specific PI service _N_FINDAB and the system is in JOG mode.

The configured reset settings are activated through the deselection:

- MD20110 $MC_RESET_MODE_MASK
- MD20150 $MC_GCODE_RESET_VALUES
- MD20151 $MC_GCODE_RESET_MODE

Supplementary conditions

- Axis couplings are not restored with the selection of the JOG retract submode.
- Tool retraction for tapping with G63 (with compensating chuck and speed-controlled spindle without encoder) is not possible. The NC/PLC interface signal DB21, … DBX377.5 (retract data available) is not set in this case.
- OEM transformations for with parallel kinematics (e.g. for hexapods) only permit manual traversing with referenced axes. In this case, restored axis positions cannot be used.
- If the tool orientation was not activated via an NC-function, but through direct programming of the orientation axes, there is no possibility of an NC-supported retraction in the direction of the tool axis.
- JOG retract can also be used in systems with NCU link. It should be noted, however, that by selecting JOG retract the current status of axis containers is not changed, and in particular old axis container states before a power failure or program abort are not restored.
- If a transformation is selected through JOG retract, restored or synchronized positions must be present for the axes involved in the transformation. The user must must ensure that axes with incremental encoders are clamped quickly enough when a power failure occurs in order to prevent a drifting away from the last position setpoint as otherwise the restored position can deviate too much from the actual position. Drive-autonomous retraction must also not be activated for these axes.

8.15 Replacing functions by subprograms

8.15.1 Overview

Function

User-specific auxiliary functions (e.g. `M101`) do not trigger any system functions. They are only output to the NC/PLC interface. The functionality of the auxiliary function must be implemented by the user/machine manufacturer in the PLC user program. A description will be provided as to how a user-specific subprogram call can be configured (replacement subprogram) instead of the output to NC/PLC interface, which is the default setting.

Function `M101` is then still programmed in the part program. However, when executing the part program, the substitute subprogram is called. Therefore, the NC replaces the function by a subprogram call. This results in the following advantages:

- When adapting to the production process, an existing, tested and proven part program can still be used, unchanged. The changes required are then shifted into the user-specific subroutines.
- The functionality can be implemented within the substitute subprogram with the full functional scope of the NC language.
- The communication overhead between NC and PLC is not required.

Functions that can be replaced

The following functions can be replaced by subprograms:

Auxiliary functions	
M	Switching functions
T	Tool selection
TCA	Tool selection independent of the tool status
D	Tool offset
DL	Additive tool offset

Spindle-related functions during active synchronous spindle coupling	
M40	Automatic gear stage change
M41 - M45	Gear stage selection 1 ... 5
SPOS	Spindle positioning
SPOSA	Spindle positioning
M19	Spindle positioning

8.15.2 Replacement of M, T/TCA and D/DL functions

8.15.2.1 Replacement of M functions

General Information

The following conditions are applicable for replacing the M functions:

- Per block only one M function is replaced.
- A block in which an M function is to be replaced, must **not** contain the following elements:
  - M98
  - Modal subroutine call
  - Subprogram return
  - Part program end
- M functions that trigger system functions must not be replaced by a subprogram (see section "Non replaceable M functions").

Parameterization

M function and subprogram

M functions and the replacement subprograms are parameterized in the following machine data:

- MD10715 $MC_M_NO_FCT_ CYCLE[<**Index**>] = <M function number>
- MD10716 $MC_M_NO_FCT_CYCLE_NAME[<**Index**>] = "<subprogram name>"

The M function and the corresponding replacement subprogram are connected through the same index.

Example: M function `M101` is replaced by subprogram SUB_M101 and M function `M102` by SUB_M102:

MD10715 $MC_M_NO_FCT_CYCLE[**0**] = 101
MD10716 $MC_M_NO_FCT_CYCLE_NAME[**0**] = "SUB_M101"

MD10715 $MC_M_NO_FCT_CYCLE[**1**] = 102
MD10716 $MC_M_NO_FCT_CYCLE_NAME[**1**] = "SUB_M102"

System variable for transferring information

For a freely selectable M function, information regarding the M function that has been replaced and additional functions (T, TCA, D, DL) for evaluation in the replacement subprogram are made available via the system variable (see Chapter "System variable (Page 676)"). The data contained in the system variables refers to the block in which the M function to be replaced is programmed.

The M function is selected with the index of machine data MD10715 $MC_M_NO_FCT_CYCLE[<Index>] in which the M function to be replaced has been parameterized:

- MD10718 $MC_ M_NO_FCT_CYCLE_PAR = <Index>

Note

For an M function replacement with transfer of information via system variable, the address extension and function value of the M function must be programmed as constant values.

Permissible programming:

- `M<function value>`
- `M=<function value>`
- `M[<address extension>]=<function value>`

Illegal programming:

- `M=<variable1>`
- `M[<variable2>]=<variable1>`

Programming

Rules for replacing M functions:

- The replacement subprogram is called at the block end
- Within the replacement subprogram, no M functions are replaced
- In an ASUB, the M function is also replaced if the ASUB was started within the replacement subprogram.

M functions that cannot be replaced

The following M functions trigger system functions as pre-defined auxiliary functions and must not be replaced by a subprogram:

- M0, ... M5
- M17, M30,
- M19
- M40, ... M45
- M98, M99 (only for MD18800 $MN_MM_EXTERN_LANGUAGE ≠ 0)

User-specific M functions parameterized via machine data must also not be replaced by a subprogram as they also trigger system functions.

Machine data	Meaning
MD10714 $MN_M_NO_FCT_EOP	M function for spindle active after RESET
MD10804 $MN_EXTERN_CHAN_M_NO_SET_INT	M function for ASUB activation (external mode)
MD10806 $MN_EXTERN_CHAN_M_NO_DISABLE_INT	M function for ASUB deactivation (external mode)
MD10814 $MN_EXTERN_M_NO_MAC_CYCLE	Macro call via M function
MD20094 $MC_SPIND_RIGID_TAPPING_M_NR	M function for switchover to controlled axis mode
MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR	M function for switchover to controlled axis mode (external mode)
MD22254 $MC_AUXFU_ASSOC_M0_VALUE	Additional M function for program stop
MD22256 $MC_AUXFU_ASSOC_M1_VALUE	Additional M function for conditional stop
MD26008 $MC_NIBBLE_PUNCH_CODE	Definition of M functions (for nibble-specific)
MD26012 $MC_PUNCHNIB_ACTIVATION	Activation of punching and nibbling functions

Note

Exception

The M function parameterized using machine data

MD22560 $MC_TOOL_CHANGE_M_CODE (Tool change with M Function)

may be replaced by a subroutine.

8.15.2.2 Replacing T/TCA and D/DL functions

Boundary conditions

For replacing functions T, TCA, D and DL, the following supplementary conditions apply:

- A maximum of one function replacement is active per block.
- A block with the function replacement must **not** contain the following elements:
  - `M98`
  - Modal subroutine call
  - Subprogram return
  - Part program end

Parameterization: Replacement subprogram

The replacement subprogram is specified function-specific in the machine data:

Function	Machine data
T	MD10717 $MN_T_NO_FCT_CYCLE_NAME
TCA	MD15710 $MN_TCA_CYCLE_NAME
D/DL	MD11717 $MN_D_NO_FCT_CYCLE_NAME

Note

It is recommended that the same subprogram is used to replace T, TCA and D/DL functions.

Parameterization: Behavior regarding D or DL function with simultaneous T function

When D or DL and T functions are simultaneously programmed in a block, the D or DL number is either transferred as parameter to the replacement subprogram or the D or DL function is executed before calling the replacement subprogram. The behavior is configurable via:

MD10719 $MN_T_NO_FCT_CYCLE_MODE (Parameterization of the T function replacement)

Bit	Value	Meaning
0	0	The D or DL number is available in the subprogram in the form of a system variable (initial state).
	1	The D or DL number is calculated directly in the block. **Note:** This function is only active if the tool change was configured with M function: MD22550 $MC_TOOL_CHANGE_MODE = 1 otherwise the D or DL values are always transferred.

System variable for transferring information

The replacement subroutine is provided with all of the information relevant to the functions programmed in the block via system variables (see Chapter "System variable (Page 676)").

The data contained in the system variables refers to the block in which the function to be replaced was programmed.

Parameterization: Time that the replacement subprogram is called

The call time of the replacement subprogram is set via:

MD10719 $MN_T_NO_FCT_CYCLE_MODE, bit 1 and bit 2

Bit 2	Bit 1	Time that the replacement subprogram is called
0	0	At the end of the block After the replacement subprogram has been executed, the interpretation is resumed with the program line following the line that triggered the replacement operation.
0	1	At block start After the replacement subprogram has been executed, the program line, which resulted in the replacement subprogram being called, is interpreted. The T address and the D or DL address and the M function for the tool change are no longer processed.
1	-	At block start and block end The replacement program is called twice.

System variable for the call time

System variable $P_SUB_STAT can be used to read whether the substitution is active, and if so, when the replacement subprogram – referred to the block – was called up:

Value	Meaning
0	Replacement not active
1	Replacement active, subprogram call is made at the block start
2	Replacement active, subprogram call is made at the block end

Example: Replacement of the T function

Parameterization	Meaning
MD22550 $MC_TOOL_CHANGE_MODE = 0	Tool change with T function
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	Name of the subprogram to replace the T function
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0	Call time: End of block

```
Programming                    Comment
N110 D1                        ; D1
N120 G90 G0 X100 Y100 Z50      ; D1 is active
N130 D2 X110 Z0 T5             ; D1 remains active
                               ; The T function is replaced at the end of the
                               ; block by
                               ; UP call MY_T_CYCLE
                               ; D2 is made available to MY_T_CYCLE in a
                                 system variable
```

A detailed example for replacement of the T function can be found in Chapter: "Examples of M/T function replacement at a tool change (Page 678)".

8.15.2.3 System variable

General Information

The replacement subprogram is provided with all of the information relevant to the functions programmed in the block (T or TCA, D or DL, M) via system variables.

Exception

D or DL number is not transferred if:

- MD10719 $MN_T_NO_FCT_CYCLE_MODE, bit 0 = 1
- MD22550 $MC_TOOL_CHANGE_MODE = 1

AND

- D or DL together with T or M function programmed in a block

⚠ CAUTION

The values provided for the replacement subprogram in the system variables are not yet effective. It is the sole responsibility of the user/machine manufacturer to resolve this by using the appropriate programming in the replacement subprogram.

System variables

System variable	Meaning
$C_M_PROG	TRUE, if the M function has been programmed
$C_M	For $C_M_PROG == TRUE, contains the value of address M We must differentiate between two cases here: • $C_M supplies the value if, for the tool change with M function, a subprogram is configured with parameter transfer: MD10715 MN_M_NO_FCT_CYCLE • If only one subprogram is configured for the addresses T and/or D/DL and if in the program the M function for the tool change is programmed together with one of the addresses to be replaced, then $C_M supplies the value: MD22560 $MC_TOOL_CHANGE_M_CODE
$C_AUX_VALUE[0]	Value of the replaced M function
$C_ME	For $C_M_PROG == TRUE, contains the value of the address extension of the M function
$C_AUX_EXT[0]	Address extension of the M function(identical to $C_ME)
$C_AUX_IS_QUICK[0]	TRUE, if the M function was programmed with quick output to the PLC
$C_T_PROG	TRUE, if the T function was programmed
$C_T	For $C_T_PROG == TRUE, contains the value of the T function
$C_TE	Contains for: • $C_T_PROG == TRUE • $C_TS_PROG == TRUE the value of the address extension of the T function
$C_TS_PROG	TRUE, if for the T or TCA replacement, a tool identifier has been programmed.
$C_TS	For $C_TS_PROG == TRUE, contains the tool identifier programmed for the T or TCA replacement
$C_TCA	TRUE, if the TCA replacement is active
$C_DUPLO_PROG	TRUE, if the duplo number of the TCA replacement has been programmed
$C_DUPLO	For $C_DUPLO_PROG == TRUE, contains the value of the programmed duplo number
$C_THNO_PROG	TRUE, if the toolholder/spindle number of the TCA replacement has been programmed
$C_THNO	For $C_THNO_PROG == TRUE, contains the value of the programmed toolholder/spindle number
$C_D_PROG	TRUE, if the D function has been programmed
$C_D	For $C_D_PROG == TRUE, contains the value of the D function
$C_DL_PROG	TRUE, if the DL function was programmed
$C_DL	For $C_DL_PROG == TRUE, contains the value of the DL function
$P_SUB_STAT	Block-related time that the replacement subprogram is called

8.15.2.4 Example: Replacement of an M function

Example 1

The function M6 is replaced by calling the subprogram "SUB_M6".

The information relevant for a tool change should be transferred using system variables.

Parameterization

Machine data	Meaning
MD10715 $MN_M_NO_FCT_CYCLE[**2**] = 6	Tool change with M6
MD10716 $MN_M_NO_FCT_CYCLE_NAME[**2**] = "SUB_M6"	Replacement subprogram for M6
MD10718 $MN_M_NO_FCT_CYCLE_PAR = **2**	Information transfer using system variables

Main program

```
Programming                          Comment
PROC MAIN
...                                  ;
N10 T1 D1 M6                         ; M6 is replaced by subroutine "SUB_M6"
                                     ;
...                                  ;
N90 M30
```

Subprogram "SUB_M6"

```
Programming                          Comment
PROC SUB_M6
N110 IF $C_T_PROG==TRUE              ; IF address T is programmed
N120 T[$C_TE]=$C_T                   ;  Execute T selection
N130 ENDIF                           ; ENDIF
N140 M[$C_ME]=6                      ; Execute tool change.
N150 IF $C_D_PROG==TRUE              ; IF address D is programmed
N160 D=$C_D                          ;  Execute D selection
N170 ENDIF                           ; ENDIF
N190 M17
```

Example 2

The new tool is prepared for changing with the T function. The tool change is only realized with function M6. The T function is replaced by calling the subprogram "MY_T_CYCLE". The D / DL number is transferred to the subprogram.

Parameterization

Parameterization	Meaning
MD22550 $MC_TOOL_CHANGE_MODE = 1	Tool change prepared with T function
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	Replacement subprogram
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0	Transfer of the D/DL number

Main program

```
Program code                    Comment
N210 D1                         ;
N220 G90 G0 X100 Y100 Z50       ; D1 is active.
N230 D2 X110 Z0 T5              ; D1 remains active, programmed D2 is transferred
                                ; to the subprogram as variable
N240 M6                         ; Execute tool change
```

Example 3

The new tool is prepared for changing with the T function. The tool change is only realized with function M6. The T function is replaced by calling the subprogram "MY_T_CYCLE". The D / DL number is **not** transferred to the subprogram.

Parameterization

Parameterization	Meaning
MD22550 $MC_TOOL_CHANGE_MODE = 1	Tool change prepared with T function
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	Replacement subprogram
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1	No transfer of the D/DL number

Main program

```
Program code                    Comment
N310 D1
N320 G90 G0 X100 Y100 Z50       ; D1 is active.
N330 D2 X110 Z0 T5              ; D2 is active and is not transferred as variable
                                ; to
                                  the replacement subprogram.
N340 M6                         ; Execute tool change.
```

Example 4

The functions T and M6 are replaced by the subprogram "MY_T_CYCLE".

The parameters are transferred to the subprogram when replacing M6.

If M6 is programmed together with D or DL in the block, the D or the DL number is also transferred as parameter to the subprogram if no transfer of the D/DL number has been parameterized:

MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1

Parameterization

Configuration	Meaning
MD22550 $MC_TOOL_CHANGE_MODE = 1	Tool change with M function
MD22560 $MC_TOOL_CHANGE_M_CODE = 6	M code for tool change
MD10715 $MC_M_NO_FCT_CYCLE[3] = 6	M function to be replaced
MD10716 $MC_M_NO_FCT_CYCLE_NAME[3] = "MY_T_CYCLE"	Replacement subprogram for the M function
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	Replacement subprogram for the T function
MD10718 $MN_M_NO_FCT_CYCLE_PAR = 3	Parameter transfer to the replacement subprogram for M6
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1	No transfer of the D/DL number

Main program

```
Program code                    Comment
N410 D1
N420 G90 G0 X100 Y100 Z50       ; D1 is active.
N330 D2 X110 Z0 T5 M6           ; D1 remains active, D2 and T5 are transferred to
                                  the M6 replacement subprogram as variable.
```

8.15.2.5 Example: Replacement of a T and D function

The functions T and D are replaced by calling the subprogram "D_T_SUB_PROG". The following should also be true for the example:

- The tool change is realized with address T.
- The subprogram is called at the start of the block.
- The tool management is not active.
- Axis B is an indexing axis with Hirth gearing.

Parameterization

Machine data	Meaning
MD11717 $MN_**D**_NO_FCT_CYCLE_NAME = "**D_T_SUB_PROG**"	Replacement subprogram for D function
MD10717 $MN_**T**_NO_FCT_CYCLE_NAME = "**D_T_SUB_PROG**"	Replacement subprogram for M function
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 'H2'	Call at block start
MD22550 $MC_TOOL_CHANGE_MODE = 0	Tool change with T function

Main program

```
Programming                          Comment
PROC MAIN
...                                  ;
N10 G01 F1000 X10 T1=5 D1            ; T and D function replaced by calling
                                     ; "D_T_SUB_PROG" at start of block
...                                  ;
N90 M30
```

Subprogram "D_T_SUB_PROG"

```
Programming                              Comment
N1000 PROC D_T_SUB_PROG DISPLOF SBLOF

N4100 IF $C_T_PROG==TRUE                 ; IF address T is programmed
N4120 POS[B]=CAC($C_T)                   ;  Approach the indexing position
N4130 T[$C_TE]=$C_T                      ;  Select tool (T selection)
N4140 ENDIF                              ; ENDIF

N4300 IF $C_T_PROG==TRUE                 ; IF address D is programmed
N4320 D=$C_D                             ;  Select offset (D selection)
N4330 ENDIF                              ; ENDIF

N4400 IF $C_DL_PROG==TRUE                ; IF address DL is programmed
N4420 D=$C_DL                            ;  Select insert offset
N4430 ENDIF                              ; ENDIF

N9999 RET
```

8.15.2.6 Behavior in the event of a conflict

Conflict case

A conflict is present if several functions are programmed in one block and the functions should be replaced with different subprograms:

- Addresses D and DL replaced with subprogram:

  MD11717 $MN_FCT_CYCLE_NAME = "**D_SUB_PROG**"
- Address T replaced with subprogram:

  MD10717 $MN_FCT_CYCLE_NAME = "**T_SUB_PROG**"
- M function M6 replaced with subprogram:

  MD10715 $MN_M_NO_FCT_CYCLE[0] = 6

  MD10716 $MN_M_NO_FCT_CYCLE_NAME[0] = "**M6_SUB_PROG**"

  MD10718 $MN_M_NO_FCT_CYCLE_PAR = 0

  MD22550 $MC_TOOL_CHANGE_MODE = 1

  MD22560 $MC_TOOL_CHANGE_M_CODE = 6

Resolution

A conflict is resolved corresponding to the following table:

The following are programmed in one program line:			Called subprogram:
D and/or DL	T or TCA	M6	
–	–	x	**M6**_SUB_PROG
–	x	–	**T**_SUB_PROG
–	x	x	**M6**_SUB_PROG
x	–	–	**D**_SUB_PROG
x	–	x	**M6**_SUB_PROG
x	x	–	**T**_SUB_PROG
x	x	x	**M6**_SUB_PROG

8.15.3 Replacement of spindle functions

8.15.3.1 General

Function

When a coupling is active the following spindle functions can be replaced for leading spindles:

- M40: Automatic gear stage change
- M41 ... M45 Programmed gear stage change
- SPOS, SPOSA and M19: Spindle positioning

Boundary conditions

- To replace a spindle function, the following conditions must be met:
  - The programmed spindle must be the leading spindle of an active coupling.
  - Leading and following spindle are located in the same channel. This is only detected if the leading spindle is located in the channel in which the coupling was closed. If the leading spindle is changed to another channel, a gear stage change or positioning of this spindle does not call the replacement subprogram.
  - A programmed gear stage change must result in a real gear stage change. For this purpose, the programmed and active gear stage must differ.
- In a block, only one spindle function can be replaced. Multiple replacements lead to the termination of the program processing. The spindle functions, which are to be replaced, must then be distributed over several blocks.

Parameterization

Spindle function

The spindle functions to be replaced by the subprogram are selected in the machine data:

MD30465 $MA_AXIS_LANG_SUB_MASK

Bit	Meaning	
0	Gear-stage change automatic (M40) and directly (M41-M45)	
	Value	Meaning
	0	No replacement
	1	Replacement through the subprogram set in MD15700 and MD15702
1	Spindle positioning with SPOS / SPOSA / M19	
	Value	Meaning
	0	No replacement
	1	Replacement through the subprogram set in MD15700 and MD15702

Subprogram: Name

The name of the replacement subprogram is entered in the machine data:

MD15700 $MN_LANG_SUB_NAME = "<subprogram name>"

Subprogram: Path

The path of the replacement subprogram is set in the machine data:

MD15702 $MN_LANG_SUB_PATH = <value>

Value	Meaning
0	**Manufacturer** cycle folder: /_N_**CMA**_DIR
1	**User** cycle folder: /_N_**CUS**_DIR
2	**Siemens** cycle folder: /_N_**CST**_DIR

System variable: Time that the replacement subprogram is called

The time that the replacement subprogram is called can be read using the system variable $P_SUB_STAT:

Value	Meaning
0	Replacement not active
1	Replacement active, subprogram call is made at the block start
2	Replacement active, subprogram call is made at the block end

Block processing

If the replacement subprogram is called at the block start, after processing the replacement subprogram, the block that initiated the call is processed. The replaced commands are no longer processed.

If the replacement subprogram is called at the block end, the block that initiated calling the replacement subprogram is first processed without the commands to be replaced. The replacement subprogram is then subsequently called.

8.15.3.2 Replacement of M40 - M45 (gear stage change)

Function

When a coupling is active, the commands for gear stage change (M40, M41 ... M45) of the leading spindle are replaced by calling a user-specific subprogram.

Parameterization

Activation

- MD30465 $MA_AXIS_LANG_SUB_MASK, bit 0 = 1

Time that the subprogram is called

- `M40`

  The time of the call cannot be set. The replacement subprogram is always called at the block start.

- `M41` ... `M45`

  The call time depends on the configured output behavior of the auxiliary function to the PLC (see below MD22080):

  – Output **before** or **during** motion: Subprogram call at the **start of the block**.

  – Output **after** motion: Subprogram call at the **end of the block**

  MD22080 $MC_AUXFU_PREDEF_SPEC[12 ... 16] (output behavior for `M41` ... `M45`)

Bit	Value	Meaning
5	1	Output of the auxiliary function **before** the motion
6	1	Output of the auxiliary function **during** the motion
7	1	Output of the auxiliary function **after** the motion

System variable to transfer information

The replacement subroutine is provided with all of the information relevant to the functions programmed in the block via system variables (see Chapter "System variable (Page 687)"). The data refer exclusively to the block, in which the function to be replaced has been programmed.

8.15.3.3 Replacement of SPOS, SPOSA, M19 (spindle positioning)

Function

When a coupling is active, the positioning commands (SPOS, SPOSA or M19) of a leading spindle are replaced by calling a user-specific subprogram (replacement subprogram).

Application example

When machining workpieces in parallel on a double-spindle machine, the spindles are coupled through a coupling factor not equal to 1. When changing the tool, they must be brought to the same position. The replacement subprogram opens the coupling, separately positions the spindles at the tool change position and then recloses the coupling.

Parameterization

Activation

- MD30465 $MA_AXIS_LANG_SUB_MASK, bit 1 = 1

Time that the replacement subprogram is called

- SPOS, SPOSA

  The time of the call cannot be set. The replacement subprogram is always called at the block start.

- M19

  The call time depends on the configured output behavior of the auxiliary function to the PLC (see below MD22080):

  – Output **before** or **during** motion: Subprogram call at the **start of the block**.

  – Output **after** motion: Subprogram call at the **end of the block**

MD22080 $MC_AUXFU_PREDEF_SPEC[9]		
Bit	**Value**	**Meaning**
5	1	Output of the auxiliary function **before** the motion
6	1	Output of the auxiliary function **during** the motion
7	1	Output of the auxiliary function **after** the motion

System variable for transferring information

The replacement subroutine is provided with all of the information relevant to the functions programmed in the block via system variables (see Chapter "System variable (Page 687)"). The data refer exclusively to the block, in which the function to be replaced has been programmed.

8.15.3.4 System variable

System variable	Meaning
$P_SUB_AXFCT	TRUE, if M40, M41 ... M45 replacement is active
$P_SUB_GEAR	Programmed or calculated gear stage Outside the replacement subprogram: Gear stage of the master spindle
$P_SUB_AUTOGEAR	TRUE, if M40 was active in the block that had initiated the replacement operation. Outside the replacement subprogram: actual setting in the interpreter
$P_SUB_LA	Contains the axis identifier of the leading spindle of the active coupling, which had triggered the replacement operation. **Note** If the variable is used outside the replacement subprogram, program processing is cancelled with an alarm.
$P_SUB_CA	Contains the axis identifier of the following spindle of the active coupling, which had triggered the replacement operation. **Note** If the variable is called outside the replacement subprogram, program processing is cancelled with an alarm.
$P_SUB_AXFCT	Contains the active replacement types corresponding to MD30465 $MA_AXIS_LANG_SUB_MASK
$P_SUB_SPOS	TRUE, if the SPOS replacement is active
$P_SUB_SPOSA	TRUE, if the SPOSA replacement is active
$P_SUB_M19	TRUE, if the M19 replacement is active
$P_SUB_SPOSIT	Contains the programmed spindle position **Note** If the variable is called outside the replacement subprogram, program processing is cancelled with an alarm.
$P_SUB_SPOSMODE	Contains the position approach mode for the prog. spindle position:

Value	Meaning
0	No change of the position approach mode
1	AC
2	IC
3	DC
4	ACP
5	ACN
6	OC
7	PC

Note: If the variable is called outside the replacement subprogram, program processing is cancelled with an alarm.
$P_SUB_STAT	Block-related time that the replacement subprogram is called

8.15.3.5 Example: Gear stage change

In the subprogram, all commands to change the gear stage `M40`, `M41` ... `M45` are replaced.

Parameterization

Machine data	Meaning
MD15700 $MN_LANG_SUB_NAME = "LANG_SUB"	Subprogram
MD15702 $MN_LANG_SUB_PATH = 0	Manufacturer's folder
MD22080 $MC_AUXFU_PREDEF_SPEC[12] = 'H21'	M41: Output prior to motion
MD22080 $MC_AUXFU_PREDEF_SPEC[13] = 'H21'	M42: Output prior to motion
MD22080 $MC_AUXFU_PREDEF_SPEC[13] = 'H21'	M43: Output prior to motion
MD22080 $MC_AUXFU_PREDEF_SPEC[15] = 'H21'	M44: Output prior to motion
MD22080 $MC_AUXFU_PREDEF_SPEC[16] = 'H21'	M45: Output prior to motion
MD30465 $MA_AXIS_LANG_SUB_MASK[AX5] = 'H0001'	Replace gear change commands

Main program

```
Programming                          Comment
PROC MAIN
N110 COUPON(S2,S1)                   ; Close the synchronous spindle coupling
N120 G01 F100 X100 S5000 M3 M43      ; Subprogram call due to M43
N130 M40                             ; Switch-on automatic gear stage change
N140 M3 S1000                        ; Subprogram call due to S1000
                                     ; and as a result initiated automatic
                                     ; Gear stage change
N9999 M30
```

Replacement subprogram "LANG_SUB", version 1

Optimized for simplicity and velocity by directly addressing the spindles (S1: Leading spindle, S2: Following spindle).

```
Programming                          Comment
N1000 PROC LANG_SUB DISPLOF SBLOF
N1100 IF($P_SUB_AXFCT ==1)           ; Replacement due to gear stage change
N1140 DELAYFSTON                     ; Start of Stop delay area
N1150 COUPOF(S2,S1)                  ; Open synchronous spindle coupling
N1160 ;gear stage change separately for leading and following spindles
N1170 M1=$P_SUB_GEAR M2=$P_SUB_GEAR
N1180 DELAYFSTON                     ; End of Stop delay area
N1190 COUPON(S2,S1)                  ; Close the synchronous spindle coupling
N1200 ENDIF
...
N9999 RET
```

Replacement subprogram "LANG_SUB", version 2

Flexibility through indirect addressing using the system variable (leading spindle: $P_SUB_LA, following spindle: $P_SUB_CA).

```
Programming                          Comment
N1000 PROC LANG_SUB DISPLOF SBLOF
N1010 DEF AXIS _LA                   ; Bit memory for leading axis / leading spindle
N1020 DEF AXIS _CA                   ; Bit memory for following axis / following
                                       spindle
N1030 DEF INT _GEAR                  ; Bit memory for gear stage

N1100 IF($P_SUB_AXFCT==1)            ; Replacement due to gear stage change
N1110 _GEAR=$P_SUB_GEAR              ; Gear stage to be activated
N1120 _LA=$P_SUB_LA                  ; Axis identifier of the leading spindle
N1130 _CA=$P_SUB_CA                  ; Axis identifier of the following spindle
N1140 DELAYFSTON                     ; Start of Stop delay area
N1150 COUPOF(_CA,_LA)                ; Open synchronous spindle coupling
N1160 ;gear stage change for leading and following spindles
N1170 M[AXTOSPI(_LA)]=_GEAR M[AXTOSPI(_CA)]=_GEAR
N1180 DELAYFSTOF                     ; End of Stop delay area
N1190 COUPON(_CA,_LA)                ; Close the synchronous spindle coupling
N1200 ENDIF
...
N9999 RET
```

8.15.3.6 Example: Spindle positioning

In the subprogram, only the replacement of commands SPOS and SPOSA is explicitly executed. Additional replacements should be supplemented in essentially the same fashion.

Parameterization

Machine data	Meaning
MD30465 $MA_AXIS_LANG_SUB_MASK[AX5] = 'H0002'	Replace positioning commands
MD22080 $MC_AUXFU_PREDEF_SPEC[9] = 'H0021'	Output of M19 to the PLC before motion

Setting Data	Meaning
SD43240 $SA_M19_SPOS[AX5] = 260	Spindle position for M19 = 260
SD43250 $SA_M19_SPOSMODE[AX5] = 4	Position approach mode for M19: "Approach in the positive direction (ACP)"

Main program

```
Programming                              Comment
PROC MAIN
...
N210 COUPON(S2,S1)                       ; Activate synchronous spindle coupling
N220 SPOS[1]=100                         ; Position leading spindle with SPOS
...
N310 G01 F1000 X100 M19                  ; Position leading spindle with M19
```

Replacement subprogram "LANG_SUB", version 1

Optimized for simplicity and velocity by directly addressing the spindles (S1: Leading spindle, S2: Following spindle).

```
Programming                              Comment
N1000 PROC LANG_SUB DISPLOF SBLOF
N2100 IF($P_SUB_AXFCT==2)
N2110 ;Replacement of SPOS/SPOSA/M19 for active synchronous spindle coupling
N2185 DELAYFSTON                         ; Start stop delay area
N2190 COUPOF(S2,S1)                      ; Open synchronous spindle coupling
N2200                                    ; Position leading and following spindles
N2210 IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)
N2220 ;SPOS and SPOSA are mapped to SPOS
N2230 CASE $P_SUB_SPOSMODE OF \
 0 GOTOF LABEL1_DC \
 1 GOTOF LABEL1_IC \
 2 GOTOF LABEL1_AC \
 3 GOTOF LABEL1_DC \
 4 GOTOF LABEL1_ACP \
 5 GOTOF LABEL1_ACN \
 DEFAULT GOTOF LABEL_ERR
LABEL1_DC: SPOS[1]=DC($P_SUB_SPOSIT) SPOS[2]=DC($P_SUB_SPOSIT)
 GOTOF LABEL1_CONT
LABEL1_IC: DELAYFSTOF
 SPOS[1]=IC($P_SUB_SPOSIT) SPOS[2]=IC($P_SUB_SPOSIT)
 DELAYFSTON
 GOTOF LABEL1_CONT
LABEL1_AC: SPOS[1]=AC($P_SUB_SPOSIT) SPOS[2]=AC($P_SUB_SPOSIT)
 GOTOF LABEL1_CONT
```

```
Programming                                  Comment
LABEL1_ACP: SPOS[1]=ACP($P_SUB_SPOSIT) SPOS[2]=ACP($P_SUB_SPOSIT)
 GOTOF LABEL1_CONT
LABEL1_ACN: SPOS[1]=ACN($P_SUB_SPOSIT) SPOS[2]=ACN($P_SUB_SPOSIT)
LABEL1_CONT:
N2250 ELSE                                   ; Positioning the spindle using M19
N2270 M1=19 M2=19                            ; Leading and following spindles
N2280 ENDIF                                  ; End replacement SPOS, SPOSA
N2285 DELAYFSTOF                             ; End of stop delay area
N2290 COUPON(S2,S1)                          ; Activate synchronous spindle coupling
N2410 ELSE
N2420 ;from here processing further replacements
...
N3300 ENDIF                                  ; End replacements
...
N9999 RET                                    ; Normal end of program
LABEL_ERR: SETAL(61000)                      ; Error has occurred
```

Replacement subprogram "LANG_SUB", version 2

Flexibility through indirect addressing using the system variable (leading spindle: $P_SUB_LA, following spindle: $P_SUB_CA).

```
Programming                                  Comment
N1000 PROC LANG_SUB DISPLOF SBLOF
N1010 DEF AXIS _LA                           ; Leading axis/spindle
N1020 DEF AXIS _CA                           ; Following axis/spindle
N1030 DEF INT _LSPI                          ; Leading spindle number (programmed
                                             ; spindle)
N1040 DEF INT _CSPI                          ; Following spindle number
...
N2100 IF($P_SUB_AXFCT==2)
N2110 ;Replacement of SPOS/SPOSA/M19 for active synchronous spindle coupling
N2120 _LA=$P_SUB_LA                          ; Axis identifier of the leading spindle
N2130 _CA=$P_SUB_CA                          ; Axis identifier of the following spindle
N2140 _LSPI=AXTOSPI(_LA)                     ; Number of the leading spindle
N2180 _CSPI=AXTOSPI(_LA)                     ; Number of the following spindle
N2185 DELAYFSTON                             ; Start stop delay area
N2190 COUPOF(_CA,_LA)                        ; Deactivate synchronous spindle coupling
N2200                                        ; Position leading and following spindle:
N2210 IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)
N2220 ;SPOS and SPOSA are mapped to SPOS
N2230 CASE $P_SUB_SPOSMODE OF
 0 GOTOF LABEL1_DC \
 1 GOTOF LABEL1_IC \
```

```
Programming                                  Comment
 2 GOTOF LABEL1_AC \
 3 GOTOF LABEL1_DC \
 4 GOTOF LABEL1_ACP \
 5 GOTOF LABEL1_ACN \
 DEFAULT GOTOF LABEL_ERR
LABEL1_DC: SPOS[_LSPI]=DC($P_SUB_SPOSIT) SPOS[_CSPI]=DC($P_SUB_SPOSIT)
 GOTOF LABEL1_CONT
LABEL1_IC: DELAYFSTOF
 SPOS[_LSPI]=IC($P_SUB_SPOSIT) SPOS[_CSPI]=IC($P_SUB_SPOSIT)
 DELAYFSTON
 GOTOF LABEL1_CONT
LABEL1_AC: SPOS[_LSPI]=AC($P_SUB_SPOSIT) SPOS[_CSPI]=AC($P_SUB_SPOSIT)
 GOTOF LABEL1_CONT
LABEL1_ACP: SPOS[_LSPI]=ACP($P_SUB_SPOSIT) POS[_CSPI]=ACP($P_SUB_SPOSIT)
 GOTOF LABEL1_CONT
LABEL1_ACN: SPOS[_LSPI]=ACN($P_SUB_SPOSIT) POS[_CSPI]=ACN($P_SUB_SPOSIT)
LABEL1_CONT:
N2250 ELSE                              ; Position the spindle using M19
N2270 M[_LSPI]=19 M[_CSPI]=19
N2280 ENDIF
N2285 DELAYFSTOF                        ; End of stop delay area
N2290 COUPON(_CA,_LA)                   ; Activate synchronous spindle coupling
N2410 ELSE
N2420 ;from here processing further replacements
...
N3300 ENDIF
...
N9999 RET                               ; Normal end of program
LABEL_ERR: SETAL(61000)                 ; Error has occurred
```

8.15.4 Properties of the subprograms

General rules

- The subprogram called when making the replacement can contain the command PROC and the attribute SBLOF and DISPLOF.
- The replacement is also made in the ISO language mode. However, the replacement subprograms are exclusively processed in the standard language mode (Siemens). There is an implicit switchover into the standard language mode. The original language mode is reselected with the return jump from the replacement subprogram.
- System variables are exclusively used to transfer information to the replacement subprogram. Transfer parameters are not possible.
- The behavior for a single block and attribute SBLOF depends on the setting in:

  MD10702 IGNORE_SINGLEBLOCK_MASK, bit 14 (prevent single-block stop)

Value	Meaning
0	The replacement subprogram behaves like a "normal" subprogram: • Return jump with M17: Stop at the end of the subprogram **Note** The output of the M function at the PLC depends on: MD20800 $MC_SPF_END_TO_VDI, bit 0 (subprogram end to PLC) - Bit 0 = 0: No output - Bit 0 = 1: M17 is output to the PLC. • Return jump with RET: No stop at the end of the replacement subprogram
1	In the block, in which the replacement subprogram is called, **only one stop is made**. Regardless of whether: • The subprogram was called at the block start and/or at the block end • Other subprograms are called in the subprogram • The subprogram is exited with M17 or RET The single block stop takes place for the replacement of M functions at the end of the replacement subprogram. For the replacement of T and D/DL functions, the time of the single block stop depends on when the subprogram is called: • Call at block start: Single block stop at the end of the block • Call at the block end: Single block stop at the end of the replacement subprogram

- For replacement subprograms with the attribute DISPLOF in the block display, the program line is displayed as actual block, which resulted in the subprogram being called.
- In the replacement subprogram, areas or the complete replacement subprogram can be protected against interruptions, such as NC stop, read-in inhibit etc., using the DELAYFSTON and DELAYFSTOF commands.
- Replacements do not occur recursively, i.e. the function that has led to the replacement subprogram call is no longer replaced if it is programmed again in the replacement subprogram.

Output of auxiliary functions to PLC

When replacing auxiliary functions, calling the replacement subprogram does not initiate that the auxiliary function is output to the PLC. The auxiliary function is only output if the auxiliary function is reprogrammed in the replacement subprogram.

Behavior during block search

The replacement subprogram is also called in the search modes "Block search with calculation" and "Block search with calculation in the program test mode" (SERUPRO). Any special features must be realized in the replacement subprogram using the system variable: $P_SEARCH, $AC_SERUPRO.

Regarding collecting actions for "search with calculation", replacement subprograms behave just like "normal" subprograms.

8.15.5 Restrictions

- Function replacements are not permitted in:
  - Synchronized actions
  - Technology cycles
- There must be no blockwise synchronized actions in front of a block that contains functions at the beginning to be replaced. See the paragraph below "Example for: Non-modal synchronized actions".
- Only the actions required for the respective replacements can be performed in the replacement subprogram.
- In a block, in which the replacement subprogram is called at the block end, the following should be observed:
  - No modal subprogram call should be active
  - No subprogram return jump should be programmed
  - No program end should be programmed

NOTICE
The controller does not monitor whether the function to be replaced has been realized in the replacement subprogram.

Example of: Non-modal synchronized actions

MD30465 $MA_AXIS_LANG_SUB_MASK, bit 0 = 1 (gear stage change)

Program code

```
...
N1000 WHENEVER $AA_IM[X2] <= $AA_IM[X1] + 0.5 DO $AA_OVR[X1]=0
N1010 G1 X100 M43
...
```

If, in block `N1010`, the function `M43` initiates that a replacement subprogram is called, machining is interrupted and an alarm is output.

8.16 Program runtime / part counter

Information on the program runtime and workpiece counter are provided to support the machine tool operator.

This information can be processed as system variables in the NC and/or PLC program. This information is also available to be displayed on the operator interface.

8.16.1 Program runtime

Function

The "program runtime" function provides internal NC timers to monitor technological processes, which can be read into the part program and into synchronized actions via the NC and channel-specific system variables.

Time since the last control power-up

The timers for measuring the time since the last control power-up are always active and can be read via NC-specific system variables:

System variable	Meaning
$AN_SETUP_TIME	Time since the last control power-up with default values ("cold restart") in minutes. Is automatically reset to "0" in each control power-up with default values.
$AN_POWERON_TIME	Time since the last normal control power-up ("warm restart") in minutes. Is automatically reset to "0" in each normal control power-up.

Program runtimes

The timers for measuring the program runtimes are:

- Available only in the mode type AUTOMATIC
- Can be read via channel-specific system variables

Some of the timers are always active, others can be activated/deactivated through MD parameterization.

The following system variables are available for the timers that are always active:

System variable	Meaning
$AC_ACT_PROG_NET_TIME	Actual net runtime of the current program in seconds Net runtime means that the time, in which the program was stopped, has been deducted. If, in the AUTOMATIC operating mode, a part program is restarted from the RESET channel state, then $AC_ACT_PROG_NET_TIME is automatically reset to "0". Additional properties: • The reset button does not reset $AC_ACT_PROG_NET_TIME back to "0" but only stops the timer. • When starting an ASUB, $AC_ACT_PROG_NET_TIME is set to "0" and also counts the runtime of the ASUB. At the end of an ASUB, it behaves just the same as for the RESET button: The timer is only held, but is not set to "0". • $AC_ACT_PROG_NET_TIME is **not** reset when starting an event-controlled program (PROG_EVENT). The program runtime is only counted further if it involves a start, M30 or a search PROG_EVENT. • The behavior of $AC_ACT_PROG_NET_TIME for GOTOS and override = 0% can be parameterized using MD27850 (refer to Section "Parameterization") **Tip:** With $AC_PROG_NET_TIME_TRIGGER, $AC_ACT_PROG_NET_TIME can be manipulated further.
$AC_OLD_PROG_NET_TIME	Net runtime in seconds of the program that has just been correctly ended "Correctly ended" means that the program was not interrupted with RESET, but instead ended properly with `M30`.) If a new program is started, $AC_OLD_PROG_NET_TIME remains unscanned, till `M30` is reached again. Additional properties: • $AC_OLD_PROG_NET_TIME is set to "0" if the currently selected program is edited. • $AC_OLD_PROG_NET_TIME is not changed at the end of an ASUB or an event-controlled program (PROG_EVENT). **Tip:** The implied copying process of $AC_ACT_PROG_NET_TIME after $AC_OLD_PROG_NET_TIME takes place only when $AC_PROG_NET_TIME_TRIGGER is not written.

System variable	Meaning	
$AC_OLD_PROG_NET_TIME_COUNT	Changes to $AC_OLD_PROG_NET_TIME After POWER ON, $AC_OLD_PROG_NET_TIME_COUNT is at "0". $AC_OLD_PROG_NET_TIME_COUNT is always increased if the control has newly written to $AC_OLD_PROG_NET_TIME. If the user terminates the running program with RESET , $AC_OLD_PROG_NET_TIME and $AC_OLD_PROG_NET_TIME_COUNT remain unchanged. With $AC_OLD_PROG_NET_TIME_COUNT it can thus be ascertained, whether $AC_OLD_PROG_NET_TIME was written. Example: If two programs running consecutively have the same runtime and were ended correctly, then the user can identify this via the changed value in $AC_OLD_PROG_NET_TIME_COUNT.	
$AC_PROG_NET_TIME_TRIGGER	Trigger for the runtime measurement Used for selective measurement of program sections i.e. by writing $AC_PROG_NET_TIME_TRIGGER in the NC program the time measurement can be enabled and disabled again: 1. $AC_PROG_NET_TIME_TRIGGER = 2 starts the measurement and in so doing sets $AC_ACT_PROG_NET_TIME to "0". 2. $AC_PROG_NET_TIME_TRIGGER = 1 ends the measurement and copies the value from $AC_ACT_PROG_NET_TIME into $AC_OLD_PROG_NET_TIME. In order to exploit all trigger options, specific values for $AC_PROG_NET_TIME_TRIGGER are filled with special functions:	
	0	Neutral state The trigger is not active.
	1	Exit Ends the measurement and copies the value from $AC_ACT_PROG_NET_TIME into $AC_OLD_PROG_NET_TIME. $AC_ACT_PROG_NET_TIME is set to "0" and then continues to run.
	2	Start Starts the measurement and in so doing sets $AC_ACT_PROG_NET_TIME to "0". $AC_OLD_PROG_NET_TIME is not changed.
	3	Stop Stops the measurement. Does not change $AC_OLD_PROG_NET_TIME and keeps $AC_ACT_PROG_NET_TIME constant until it resumes
	4	Continue The measurement is resumed, i.e. a measurement that was previously stopped is continued. $AC_ACT_PROG_NET_TIME continues to run. $AC_OLD_PROG_NET_TIME is not changed.
All system variables are reset to 0 as a result of POWER ON!		

Note

Residual time for a workpiece

If the same workpieces are produced one after the other, then from the timer values:

- Processing time for the last workpiece produced (see $AC_OLD_PROG_NET_TIME)

and

- Current processing time (see $AC_ACT_PROG_NET_TIME)

the remaining residual time for a workpiece can be determined.

The residual time is displayed on the user interface in addition to the current processing time.

NOTICE

Using STOPRE

The system variables $AC_OLD_PROG_NET_TIME and $AC_OLD_PROG_NET_TIME_COUNT do not generate any implicit preprocessing stop. This is uncritical when used in the part program if the value of the system variables comes from the previous program run. However, if the trigger for the runtime measurement ($AC_PROG_NET_TIME_TRIGGER) is written very frequently and as a result $AC_OLD_PROG_NET_TIME changes very frequently, then an explicit `STOPRE` should be used in the part program.

The following system variables are available for the timers that are activated/deactivated through MD parameterization:

System variable	Meaning
$AC_OPERATING_TIME	Total runtime of NC programs in Automatic mode (in s) In the automatic mode, the runtimes of all programs between NC start and end of program / reset are summed up. The default is not to count in NC STOP and override = 0%. Continued counting can be activated at an override of 0% via MD27860. The value is automatically reset to "0" every time the control powers up.
$AC_CYCLE_TIME	Runtime of the selected NC program (in seconds) The runtime between NC Start and End of program / NC-Reset is measured in the selected NC program. The default is not to count in NC STOP and override = 0%. Continued counting can be activated at an override of 0% via MD27860. The value is automatically reset to "0" every time a new NC program starts up. MD27860 can be set to ensure that this value will be deleted even if there is a jump to the beginning of the program with GOTOS or in the event of ASUBs and PROG_EVENTs starting.
$AC_CUTTING_TIME	Processing time in seconds The runtime of the path axes (at least one is active) is measured in all NC programs between NC start and end of program/NC reset without rapid traverse active. MD27860 can be used to set whether measuring should only be conducted when the tool is active or independent of the tool state. The measurement is interrupted when a dwell time is active. The value is automatically reset to "0" every time the control powers up with default values.

Activation/deactivation

The timer that can be activated is switched-in/switched-out using machine data:

MD27860 $MC_PROCESSTIMER_MODE, Bit 0-2

Bit	Value	Meaning
0	0	Timer for $AC_OPERATING_TIME not active.
	1	Timer for $AC_OPERATING_TIME active.
1	0	Timer for $AC_CYCLE_TIME not active.
	1	Timer for $AC_CYCLE_TIME active.
2	0	Timer for $AC_CUTTING_TIME not active.
	1	Timer for $AC_CUTTING_TIME active.

Parameterization

Behavior of the timer that is always active

The behavior of the timer that is always active for GOTOS and override = 0% is set using machine data:

MD27850 $MC_PROG_NET_TIMER_MODE

Bit	Value	Meaning
0	0	$AC_ACT_PROG_NET_TIME is not reset to "0" in case of a jump with GOTOS to the program start (initial setting).
	1	With a jump with GOTOS to the start of the program, $AC_ACT_PROG_NET_TIME is reset to "0", the value is first saved in $AC_OLD_PROG_NET_TIME and the program counter $AC_OLD_PROG_NET_TIME_COUNT is incremented.
1	0	For override = 0%, $AC_ACT_PROG_NET_TIME is not increased. This means that the program runtime is measured without the time for which the override was set to "0" (basic setting).
	1	Also for override = 0%, $AC_ACT_PROG_NET_TIME is increased. This means that the program runtime is measured with the time for which the override was set to "0".

Behavior of the timer that can be activated

The behavior of the timer that can be activated for certain functions (e.g. test run feedrate, program test) is set using machine data:

MD27860 $MC_PROCESSTIMER_MODE

Bit	Value	Meaning
4	0	No measurement during active dry run feedrate.
	1	Measurement during active dry run feedrate.
5	0	No measurement during program test.
	1	Measurement during program test.
6	Only for bit 1 = 1 (timer for $AC_CYCLE_TIME is active)	
	0	$AC_CYCLE_TIME is reset to "0" also in case of Start through ASUB and PROG_EVENTs.
	1	$AC_CYCLE_TIME is not reset to "0" in case of Start through ASUB and PROG_EVENTs.
7	Only for bit 2 = 1 (timer for $AC_CUTTING_TIME is active)	
	0	Timer for $AC_CUTTING_TIME counts only for the active tool.
	1	Timer for $AC_CUTTING_TIME counts independent of the tool.
8	Only for bit 1 = 1 (timer for $AC_CYCLE_TIME is active)	
	0	$AC_CYCLE_TIME is not reset to "0" in case of a jump with GOTOS to the program start.
	1	$AC_CYCLE_TIME is reset to "0" in case of a jump with GOTOS to the program start.
9	Only for bit 0, 1 = 1 (timer for $AC_OPERATING_TIME and $AC_CYCLE_TIME are active)	
	0	Counting of the program runtime is not continued at an override of 0%.
	1	Counting of the program runtime continues at an override of 0%.

Supplementary conditions

- **Block search**

  No program runtimes are determined through block searches.

- **REPOS**

  The duration of a REPOS process is added to the current processing time ($AC_ACT_PROG_NET_TIME).

Examples

Example 1: Parameterization of the runtime measurement via MD27860

- Activating the runtime measurement for the active NC program and hence no measurement in case of active dry run feedrate and program test:

  MD27860 $MC_PROCESSTIMER_MODE = 'H2'

- Activating the measurement for the tool action time and measurement also with active dry run feedrate and program test.

  MD27860 $MC_PROCESSTIMER_MODE = 'H34'

- Activating the measurement for the total runtime and the processing time with an active tool, including measurement with a program test:

  MD27860 $MC_PROCESSTIMER_MODE = 'H25'

- Activating the measurement for the total runtime and the machining time (independent of the tool), including measurement with a program test:

  MD27860 $MC_PROCESSTIMER_MODE = 'Ha5'

- Activating the measurement for the processing time with an active tool, including measurements at an override = 0%, but not with a trial run feed active:

  MD27860 $MC_PROCESSTIMER_MODE = 'H22'

Example 2: Measuring the duration of "mySubProgrammA"

Program code

```
...
N50 DO $AC_PROG_NET_TIME_TRIGGER=2
N60 FOR ii= 0 TO 300
N70 mySubProgrammA
N80 DO $AC_PROG_NET_TIME_TRIGGER=1
N95 ENDFOR
N97 mySubProgrammB
N98 M30
```

After the program has processed line N80, the net runtime of "mySubProgrammA" is located in $AC_OLD_PROG_NET_TIME.

The value from $AC_OLD_PROG_NET_TIME:

- Is kept beyond M30.
- Is updated each time the loop is run through.

Example 3: Measuring the duration of "mySubProgrammA" and "mySubProgrammC"

```
Program code
N10 DO $AC_PROG_NET_TIME_TRIGGER=2
N20 mySubProgrammA
N30 DO $AC_PROG_NET_TIME_TRIGGER=3
N40 mySubProgrammB
N50 DO $AC_PROG_NET_TIME_TRIGGER=4
N60 mySubProgrammC
N70 DO $AC_PROG_NET_TIME_TRIGGER=1
N80 mySubProgrammD
N90 M30
```

8.16.2 Workpiece counter

Function

Various counters with a range of values from 0 to 999,999,999 are available with the "Workpiece counter" function in the form of channel-specific system variables. Read and write access to the system variables is possible.

The following channel-specific machine data can be used to control counter activation, counter reset timing and the counting algorithm.

System variables for workpiece counting

System variable	Meaning
$AC_REQUIRED_PARTS	Number of workpieces to be produced (setpoint number of workpieces) In this counter the number of workpieces at which the actual workpiece count ($AC_ACTUAL_PARTS) will be reset to "0" can be defined. MD27880 can be used to activate the generation of the display alarm: "Channel %1 workpiece target = %2 reached" and of the channel-specific NC/PLC interface signal: DB21, DBX317.1 (workpiece target reached) .
$AC_TOTAL_PARTS	Total number of completed workpieces (actual workpiece total) This counter specifies the total number of workpieces produced since the start time. The value is only automatically reset to "0" when the control runs up with default values.
$AC_ACTUAL_PARTS	Number of completed workpieces (actual workpiece total) This counter registers the total number of workpieces produced since the start time. On condition that $AC_REQUIRED_PARTS > 0, the counter is automatically reset to "0" when the required number of workpieces ($AC_REQUIRED_PARTS) is reached.
$AC_SPECIAL_PARTS	Number of workpieces selected by the user This counter supports user-specific workpiece counts. An alarm can be defined to be output when the setpoint number of workpieces is reached ($AC_REQUIRED_PARTS). Users must reset the counter themselves.

Note

All workpiece counters are set to "0" when the control runs up with default values and can be read and written independent of their activation.

Activation

The workpiece counter is activated with the machine data:

MD27880 $MC_PART_COUNTER (activation of workpiece counters)

Bit	Value	Meaning
0	1	$AC_REQUIRED_PARTS is active
1	0	Alarm / signal output in case of: $AC_ACTUAL_PARTS = $AC_REQUIRED_PARTS
	1	Alarm / signal output in case of: $AC_SPECIAL_PARTS = $AC_REQUIRED_PARTS
4	1	$AC_TOTAL_PARTS is active.
5	0	$AC_TOTAL_PARTS is incremented by the value "1" through M02/M30.
	1	$AC_TOTAL_PARTS is incremented by the value "1" through the M command defined with MD27882[**0**].
6	0	$AC_TOTAL_PARTS is also active for program test / block search.
7	1	$AC_TOTAL_PARTS is incremented by the value "1" upon a jump back with GOTOS.
8	1	$AC_ACTUAL_PARTS is active
9	0	$AC_ACTUAL_PARTS is incremented by the value "1" through M02/M30.
	1	$AC_ACTUAL_PARTS is incremented by the value "1" through the M command defined with MD27882[**1**].
10	0	$AC_ACTUAL_PARTS is also active for program test / block search.
11	1	$AC_ACTUAL_PARTS is incremented by the value "1" upon a jump back with GOTOS.
12	1	$AC_SPECIAL_PARTS is active.
13	0	$AC_SPECIAL_PARTS is incremented by the value "1" through M02/M30.
	1	$AC_SPECIAL_PARTS is incremented by the value 1 through the M command defined with MD27882[**2**].
14	0	$AC_SPECIAL_PARTS is also active for program test / block search.
15	1	$AC_SPECIAL_PARTS is incremented by the value "1" upon a jump back with GOTOS.

Workpiece counting with user-defined M command

If the corresponding bit is set in MD27880, then the count pulse can be triggered via a user-defined M command parameterized via the following machine data instead of via the end of program M2/M30.

MD27882 $MC_PART_COUNTER_MCODE[<n>] (workpiece counting with user-defined M command)

<n>	Meaning
0	MD27882[**0**] defines the M command in which $AC_**TOTAL**_PARTS is incremented.
1	MD27882[**1**] defines the M command in which $AC_**ACTUAL**_PARTS is incremented.
2	MD27882[**2**] defines the M command in which $AC_**SPECIAL**_PARTS is incremented.

The respective workpiece counter is incremented by "1", when a user-defined M command is called.

Supplementary conditions

- **Mode change / NC RESET**

  The counters are not affected by a mode change or NC RESET.

- **$AC_REQUIRED_PARTS ≤ 0**

  Where $AC_REQUIRED_PARTS ≤ 0 and MD27880 $MC_PART_COUNTER, bit 0 = 1, the counting procedure and the identity comparison set with MD27880 are **not** conducted for all the active counters.

Examples

- **Activation of the workpiece counter $AC_REQUIRED_PARTS:**

  MD27880 $MC_PART_COUNTER = 'H3'

  – $AC_REQUIRED_PARTS is active

  – Display alarm at: $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS

- **Activation of the workpiece counter $AC_TOTAL_PARTS:**

  MD27880 $MC_PART_COUNTER = 'H10'

  MD27882 $MC_PART_COUNTER_MCODE[0] = 80

  – $AC_TOTAL_PARTS is active; the counter is incremented by the value 1 with each `M02`.

  – $MC_PART_COUNTER_MCODE[0] has no significance.

- **Activation of the workpiece counter $AC_ACTUAL_PARTS:**

  MD27880 $MC_PART_COUNTER = 'H300'

  MD27882 $MC_PART_COUNTER_MCODE[1] = 17

  – $AC_TOTAL_PARTS is active; the counter is incremented by a value of "1" with each `M17`.

- **Activation of the workpiece counter $AC_SPECIAL_PARTS:**

  MD27880 $MC_PART_COUNTER = 'H3000'

  MD27882 $MC_PART_COUNTER_MCODE[2] = 77

  – $AC_SPECIAL_PARTS is active.

  – The following takes place with every `M77`: $AC_SPECIAL_PARTS + 1

- **Deactivation of the workpiece counter $AC_ACTUAL_PARTS:**

  MD27880 $MC_PART_COUNTER = 'H200'

  MD27882 $MC_PART_COUNTER_MCODE[1] = 50

  – $AC_ACTUAL_PARTS is inactive

- **Activation of all counters:**

  MD27880 $MC_PART_COUNTER = 'H3313'

  MD27882 $MC_PART_COUNTER_MCODE[0] = 80

  MD27882 $MC_PART_COUNTER_MCODE[1] = 17

  MD27882 $MC_PART_COUNTER_MCODE[2] = 77

  – $AC_REQUIRED_PARTS is active

  – Display alarm at: $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS

  – $AC_TOTAL_PARTS is active; the counter is incremented by the value 1 with each M02.

  – $MC_PART_COUNTER_MCODE[0] has no significance.

  – $AC_ACTUAL_PARTS is active; the counter is incremented by a value of "1" with each M17.

  – $AC_SPECIAL_PARTS is active; the counter is incremented by a value of "1" with each M77.

- **Workpiece counter $AC_ACTUAL_PARTS is not processed during the program test / block search:**

  MD27880 $MC_PART_COUNTER = 'H700'

  MD27882 $MC_PART_COUNTER_MCODE[1] = 75

  – $AC_ACTUAL_PARTS is active; the counter is incremented by a value of "1" with each M75, apart from during the program test and search.

- **Cancellation of the count modes in the MD27880 $MC_PART_COUNTER with bit 0 = 1:**

  MD27882 $MC_PART_COUNTER_MCODE[0] = 41

  MD27882 $MC_PART_COUNTER_MCODE[1] = 42

  MD27882 $MC_PART_COUNTER_MCODE[2] = 43

```
Program code                          Comment
...
N100 $AC_REQUIRED_PARTS=-10           ; Value < 0 ⇒ stop counting.
N200 M41 M43                          ; Not counting.
N300 M42
...
N500 $AC_REQUIRED_PARTS=52            ; Value > 0: Counting in accordance with MD27880
                                        activated.
N501 M43                              ; Counting.
N502 M42 M41                          ; Counting.
...
```

8.17 Data lists

8.17.1 Machine data

8.17.1.1 General machine data

Displaying machine data

Number	Identifier: $MM_	Description
SINUMERIK Operate		
9421	MA_AXES_SHOW_GEO_FIRST	Display geo axes of channel first
9422	MA_PRESET_MODE	PRESET / basic offset in JOG.
9423	MA_MAX_SKP_LEVEL	Maximum number of skip levels

NC-specific machine data

Number	Identifier: $MN_	Description
10010	ASSIGN_CHAN_TO_MODE_GROUP	Channel valid in mode group
10280	PROG_FUNCTION_MASK	Comparison commands ">" and "<"
10700	PREPROCESSING_LEVEL	Program preprocessing level
10702	IGNORE_SINGLEBLOCK_MASK	Prevent single-block stop
10707	PROG_TEST_MASK	Program test modes
10708	SERUPRO_MASK	Block change modes
10710	PROG_SD_RESET_SAVE_TAB	Setting data to be updated
10711	NC_LANGUAGE_CONFIGURATION	Manner of handling the languages, whose related option or function is not activated.
10713	M_NO_FCT_STOPRE	M function with preprocessing stop
10715	M_NO_FCT_CYCLE	M function to be replaced by subroutine
10716	M_NO_FCT_CYCLE_NAME	Subroutine name for M function replacement
10717	T_NO_FCT_CYCLE_NAME	Name of the tool change cycle for T function replacement
10718	M_NO_FCT_CYCLE_PAR	M function replacement with parameters
10719	T_NO_FCT_CYCLE_MODE	Parameter assignment for T function replacement
11450	SEARCH_RUN_MODE	Block search parameter settings
11470	REPOS_MODE_MASK	Repositioning properties
11600	BAG_MASK	Mode group response to ASUB
11602	ASUP_START_MASK	Ignore stop conditions for ASUB
11604	ASUP_START_PRIO_LEVEL	Priorities, starting from which $MN_ASUP_START_MASK is effective

Number	Identifier: $MN_	Description
11610	ASUP_EDITABLE	Activating a user-specific ASUB program
11612	ASUP_EDIT_PROTECTION_LEVEL	Protection level of user-specific ASUB program
11620	PROG_EVENT_NAME	Program name for PROG_EVENT
11717	D_NO_FCT_CYCLE_NAME	Subroutine name for D function replacement
15700	LANG_SUB_NAME	Name for replacement subroutine
15702	LANG_SUB_PATH	Call path for replacement subroutine
17200	GMMC_INFO_NO_UNIT	Global HMI info (without physical unit)
17201	GMMC_INFO_NO_UNIT_STATUS	Global HMI status info (without physical unit)
18360	MM_EXT_PROG_BUFFER_SIZE	FIFO buffer size for execution from external source (DRAM)
18362	MM_EXT_PROG_NUM	Number of program levels that can be processed simultaneously from external

8.17.1.2 Channel-specific machine data

Basic machine data

Number	Identifier: $MC_	Description
20000	CHAN_NAME	Channel name
20050	AXCONF_GEOAX_ASSIGN_TAB	Assignment of geometry axis to channel axis
20060	AXCONF_GEOAX_NAME_TAB	Geometry axis name in channel
20070	AXCONF_MACHAX_USED	Machine axis number valid in channel
20080	AXCONF_CHANAX_NAME_TAB	Channel axis name in channel [channel axis no.]: 0...7
20090	SPIND_DEF_MASTER_SPIND	Initial setting of master spindle in channel
20100	DIAMETER_AX_DEF	Geometry axis with transverse axis function
20106	PROG_EVENT_IGN_SINGLEBLOCK	Prog events ignore the single block
20107	PROG_EVENT_IGN_INHIBIT	Prog events ignore the read-in disable
20108	PROG_EVENT_MASK	Event-driven program calls
20109	PROG_EVENT_MASK_PROPERTIES	Prog events properties
20114	MODESWITCH_MASK	Setting for REPOS
20116	IGNORE_INHIBIT_ASUP	Execute user ASUBs completely in spite of read-in disable
20117	IGNORE_SINGLEBLOCK_ASUP	Process user ASUBs completely in spite of single-block processing
20160	CUBIC_SPLINE_BLOCKS	Number of blocks for C spline
20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of NC block for compression
20191	IGN_PROG_STATE_ASUP	Do not display the execution of the interrupt routine on the operator panel

Number	Identifier: $MC_	Description
20192	PROG_EVENT_IGN_PROG_STATE	Do not display the execution of the program events on the operator panel
20193	PROG_EVENT_IGN_STOP	Prog events ignore the Stop key
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Maximum value for DISC
20230	CUTCOM_CURVE_INSERT_LIMIT	Maximum angle for intersection calculation with tool radius compensation
20240	CUTCOM_MAXNUM_CHECK_BLOCKS	Blocks for predictive contour calculation with tool radius compensation
20250	CUTCOM_MAXNUM_DUMMY_BLOCKS	Maximum number of blocks without traversing motion for TRC
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20400	LOOKAH_USE_VELO_NEXT_BLOCK	Look Ahead to programmed following block velocity
20430	LOOKAH_NUM_OVR_POINTS	Number of override corner values for Look Ahead
20440	LOOKAH_OVR_POINTS	Override switch points for LookAhead
20500	CONST_VELO_MIN_TIME	Minimum time with constant velocity
20600	MAX_PATH_JERK	Path-related maximum jerk
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid motions
20700	REFP_NC_START_LOCK	NC start disable without reference point
20750	ALLOW_GO_IN_G96	G0 logic for G96, G961
20800	SPF_END_TO_VDI	Subprogram end to PLC
21000	CIRCLE_ERROR_CONST	Circle end point monitoring constant
21010	CIRCLE_ERROR_FACTOR	Circle end point monitoring factor
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21110	X_AXIS_IN_OLD_X_Z_PLANE	Coordinate system for automatic Frame definition
21200	LIFTFAST_DIST	Traversing path for fast retraction from the contour
21210	SETINT_ASSIGN_FASTIN	HW assignment of the ext. NCK input byte for NC program interrupt:
21202	LIFTFAST_WITH_MIRROR	Lift fast with mirror

Block search

Number	Identifier: $MC_	Description
20128	COLLECT_TOOL_CHANGE	Tool change commands to the PLC after block search
22600	SERUPRO_SPEED_MODE	Velocity with block search type 5
22601	SERUPRO_SPEED_FACTOR	Velocity factor for block search type 5
22621	ENABLE_START_MODE_MASK_PRT	Switches MD22620: START_MODE_MASK_ PRT for SERUPRO search run
22622	DISABLE_PLC_START	Allow part program start via PLC
22680	AUTO_IPTR_LOCK	Disable interrupt pointer

Reset response

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Initial setting after RESET / parts program end
20112	START_MODE_MASK	Basic setting for NC start after part program start
20118	GEOAX_CHANGE_RESET	Allow automatic geometry axis change
20120	TOOL_RESET_VALUE	Tool length compensation when powering-up (RESET / part program end)
20121	TOOL_PRESEL_RESET_VALUE	Preselected tool on RESET
20130	CUTTING_EDGE_RESET_VALUE	Tool cutting edge length compensation when powering-up (RESET / part program end)
20140	TRAFO_RESET_VALUE	Transformation data set when powering-up (RESET / part program end)
20150	GCODE_RESET_VALUES	Initial setting of the G groups
20152	GCODE_RESET_MODE	Reset behavior of G groups
20156	MAXNUM_GCODES_EXT	Reset behavior of the external G groups
22620	START_MODE_MASK_PRT	Initial setting at special NC Start after power-up and at RESET

Auxiliary function settings

Number	Identifier: $MC_	Description
22000	AUXFU_ASSIGN_GROUP	Auxiliary function group
22010	AUXFU_ASSIGN_TYPE	Auxiliary function type
22020	AUXFU_ASSIGN_EXTENSION	Auxiliary function extension
22030	AUXFU_ASSIGN_VALUE	Auxiliary function value
22200	AUXFU_M_SYNC_TYPE	Output timing of M functions
22210	AUXFU_S_SYNC_TYPE	Output timing of S functions
22220	AUXFU_T_SYNC_TYPE	Output timing of T functions
22230	AUXFU_H_SYNC_TYPE	Output timing of H functions
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions
22250	AUXFU_D_SYNC_TYPE	Output timing of D functions
22400	S_VALUES_ACTIVE_AFTER_RESET	S function active after RESET
22410	F_VALUES_ACTIVE_AFTER_RESET	F function active after RESET
22510	GCODE_GROUPS_TO_PLC	G codes that are output to the NCK/PLC interface on block change/RESET
22550	TOOL_CHANGE_MODE	New tool offset for M function
22560	TOOL_CHANGE_M_CODE	M function for tool change

Memory settings

Number	Identifier: $MC_	Description
27900	REORG_LOG_LIMIT	Percentage of IPO buffer for log file enable
28000	MM_REORG_LOG_FILE_MEM	Memory size for REORG (DRAM)
28010	MM_NUM_REORG_LUD_MODULES	Number of blocks for local user variables for REORG
28020	MM_NUM_LUD_NAMES_TOTAL	Number of local user variables (DRAM)
28040	MM_LUD_VALUES_MEM	Memory size for local user variables (DRAM)
28050	MM_NUM_R_PARAM	Number of channelspecific R parameters (SRAM)
28060	MM_IPO_BUFFER_SIZE	Number of NC blocks in IPO buffer (DRAM)
28070	MM_NUM_BLOCKS_IN_PREP	Number of blocks for block preparation (DRAM)
28080	MM_NUM_USER_FRAMES	Number of settable Frames (SRAM)
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of block elements for compile cycles (DRAM)
28100	MM_NUM_CC_BLOCK_USER_MEM	Size of block memory for compile cycles (DRAM)
28400	MM_ABSBLOCK	Activating basis blocks with absolute values
28402	MM_ABSBLOCK_BUFFER_CONF	Dimension size of upload buffer
28500	MM_PREP_TASK_STACK_SIZE	Stack size of preparation task (DRAM)

Program runtime and workpiece counter

Number	Identifier: $MC_	Description
27860	PROCESSTIMER_MODE	Activate the runtime measurement
27880	PART_COUNTER	Activate the workpiece counter
27882	PART_COUNTER_MCODE[ ]	Workpiece counting via M command

8.17.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30465	AXIS_LANG_SUB_MASK	Substitution of NC language commands
30550	AXCONF_ASSIGN_MASTER_CHAN	Reset position of channel for axis change
30600	FIX_POINT_POS	Fixed value positions of axes with G75
33100	COMPRESS_POS_TOL	Maximum deviation with compensation

8.17.2 Setting data

8.17.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42000	THREAD_START_ANGLE	Start angle for thread
42010	THREAD_RAMP_DISP	Acceleration behavior of axis when thread cutting
42100	DRY_RUN_FEED	Dry run feedrate
42200	SINGLEBLOCK2_STOPRE	Activate debug mode for SBL2
42444	TARGET_BLOCK_INCR_PROG	Continuation mode after block search with calculation
42700	EXT_PROG_PATH	Program path for external subroutine call EXTCALL
42750	ABSBLOCK_ENABLE	Enable basic block display
42990	MAX_BLOCKS_IN_IPOBUFFER	Maximum number of blocks in the interpolation buffer

8.17.3 Signals

8.17.3.1 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Emergency stop	DB10.DBX56.1	DB2600.DBX0.1

8.17.3.2 Signals to mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
AUTOMATIC mode	DB11.DBX0.0	DB3000.DBX0.0
MDA mode	DB11.DBX0.1	DB3000.DBX0.1
JOG mode	DB11.DBX0.2	DB3000.DBX0.2
Mode change disable	DB11.DBX0.4	DB3000.DBX0.4
Mode group Stop	DB11.DBX0.5	-
Mode group Stop axes plus spindles	DB11.DBX0.6	-
Mode group RESET	DB11.DBX0.7	DB3000.DBX0.7
Machine function teach in	DB11.DBX1.0	DB3000.DBX1.0
Machine function REPOS	DB11.DBX1.1	-
Machine function REF	DB11.DBX1.2	DB3000.DBX1.2

8.17.3.3 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Selected mode AUTOMATIC	DB11.DBX4.0	-
Selected mode MDI	DB11.DBX4.1	-
Selected JOG mode	DB11.DBX4.2	-
Selected machine function teach in	DB11.DBX5.0	-
Selected machine function REPOS	DB11.DBX5.1	-
Selected machine function REF	DB11.DBX5.2	-
Active mode AUTOMATIC	DB11.DBX6.0	DB3100.DBX0.0
Active mode MDI	DB11.DBX6.1	DB3100.DBX0.1
Active mode JOG	DB11.DBX6.2	DB3100.DBX0.2

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode group ready	DB11.DBX6.3	DB3100.DBX0.3
Mode group has been reset	DB11.DBX6.4	-
NCK internal JOG active	DB11.DBX6.5	-
All channels in the reset state	DB11.DBX6.7	-
Active machine function teach in	DB11.DBX7.0	DB3100.DBX1.0
Active machine function REPOS	DB11.DBX7.1	-
Active machine function REF	DB11.DBX7.2	DB3100.DBX1.2

8.17.3.4 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate DRF	DB21, ...DBX0.3	DB3200.DBX0.3
Activate single block	DB21, ...DBX0.4	DB3200.DBX0.4
Activate M01	DB21, ...DBX0.5	DB3200.DBX0.5
Activate dry run feed	DB21, ...DBX0.6	DB3200.DBX0.6
PLC action completed	DB21, ...DBX1.6	-
Activate program test	DB21, ...DBX1.7	DB3200.DBX1.7
Skip block levels: /0 to /7	DB21, ...DBX2.0-7	DB3200.DBX2.0-7
Read-in disable	DB21, ...DBX6.1	DB3200.DBX6.1
Program level abort	DB21, ...DBX6.4	DB3200.DBX6.4
NC start disable	DB21, ...DBX7.0	DB3200.DBX7.0
NC start	DB21, ...DBX7.1	DB3200.DBX7.1
NC Stop at block limit	DB21, ...DBX7.2	DB3200.DBX7.2
NC stop	DB21, ...DBX7.3	DB3200.DBX7.3
NC Stop axes plus spindles	DB21, ...DBX7.4	DB3200.DBX7.4
Reset	DB21, ...DBX7.7	-
REPOSPATHMODE	DB21, ...DBX31.0-2	-
REPOSMODEEDGE	DB21, ...DBX31.4	-

8.17.3.5 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
DRF selected	DB21, ...DBX24.3	DB1700.DBX0.3
Select NCK associated M01	DB21, ...DBX24.4	-
M01 selected	DB21, ...DBX24.5	DB1700.DBX0.5
Dry run feedrate selected	DB21, ...DBX24.6	DB1700.DBX0.6
REPOSPATHMODE 0 - 2	DB21, ...DBX25.0-2	-
Feedrate override selected for rapid traverse	DB21, ...DBX25.3	DB1700.DBX1.3
REPOS MODE EDGE	DB21, ...DBX25.4	-

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Program test selected	DB21, ...DBX25.7	DB1700.DBX1.7
Skip block selected /0 - /7	DB21, ...DBX26.0-7	DB1700.DBX2.0-7
Skip block selected /8	DB21, ...DBX27.0	DB1700.DBX3.0
Skip block selected /9	DB21, ...DBX27.1	DB1700.DBX3.1
REPOSPATHMODE 0 - 2	DB21, ...DBX31.0-2	-
REPOS MODE EDGE	DB21, ...DBX31.4	-
Skip block active /8	DB21, ...DBX31.6	DB3200.DBX15.6
Skip block active /9	DB21, ...DBX31.7	DB3200.DBX15.7
Execution from external active	DB21, ...DBX32.0	DB3300.DBX0.0
Action block active	DB21, ...DBX32.3	DB3300.DBX0.3
Approach block active	DB21, ...DBX32.4	DB3300.DBX0.4
M0/M1 active	DB21, ...DBX32.5	DB3300.DBX0.5
Last action block active	DB21, ...DBX32.6	DB3300.DBX0.6
Block search active	DB21, ...DBX33.4	DB3300.DBX1.4
M02/M30 active	DB21, ...DBX33.5	DB3300.DBX1.5
Transformation active	DB21, ...DBX33.6	DB3300.DBX1.6
Program test active	DB21, ...DBX33.7	DB3300.DBX1.7
Program state: Running	DB21, ...DBX35.0	DB3300.DBX3.0
Program state: Waiting	DB21, ...DBX35.1	DB3300.DBX3.1
Program state: Stopped	DB21, ...DBX35.2	DB3300.DBX3.2
Program state: Interrupted	DB21, ...DBX35.3	DB3300.DBX3.3
Program state: Aborted	DB21, ...DBX35.4	DB3300.DBX3.4
Channel state: Active	DB21, ...DBX35.5	DB3300.DBX3.5
Channel state: Interrupted	DB21, ...DBX35.6	DB3300.DBX3.6
Channel state: Reset	DB21, ...DBX35.7	DB3300.DBX3.7
Interrupt handling active	DB21, ...DBX36.4	-
Channel is ready	DB21, ...DBX36.5	-
Read-in enable is ignored	DB21, ...DBX37.6	-
Stop at the end of block with SBL is suppressed	DB21, ...DBX37.7	-
Number of the active G function of G function group 1 – n (8 bit int)	DB21, ...DBB208-271	DB3500.DBB0-63
Workpiece setpoint reached	DB21, ...DBX317.1	DB3300.DBX4001.1
ASUB is stopped	DB21, ...DBX318.0	DB3300.DBX4002.0
Block search via program test is active	DB21, ...DBX318.1	-
REPOS MODE EDGEACKN	DB21, ...DBX319.0	-
Repos Path Mode Ackn: 0 - 2	DB21, ...DBX319.1-3	-
Repos DEFERAL Chan	DB21, ...DBX319.5	-
Display of the triggering event in case of event-driven program call	DB21, ...DBX376.0-7	DB3300.DBB4004
ASUB is active	DB21, ...DBX378.0	DB3300.DBB4006.0
ASUB with suppressed display update is active	DB21, ...DBX378.1	DB3300.DBB4006.1

8.17.3.6 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
REPOSDELAY	DB31, ...DBX10.0	-

8.17.3.7 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
REPOS offset	DB31, ...DBX70.0	-
REPOS offset valid	DB31, ...DBX70.1	-
REPOS Delay Ackn	DB31, ...DBX70.2	-
REPOSDELAY	DB31, ...DBX72.0	-
Path axis	DB31, ...DBX76.4	DB390x.DBX1002.4

K2: Axis Types, Coordinate Systems, Frames 9

9.1 Brief description

9.1.1 Axes

Machine axes

Machine axes are the axes that actually exist on a machine tool.

Channel axes

Every geometry axis and every special axis is assigned to a channel and, therefore, a channel axis. Geometry axes and additional axes are always traversed in "their" channel.

Geometry axes

The three geometry axes always make up a fictitious rectangular coordinate system, the basic coordinate system (BCS).

By using FRAMES (offset, rotation, scaling, mirroring), it is possible to image geometry axes of the workpiece coordinate system (WCS) on the BCS.

Special axes

In contrast to geometry axes, no geometrical relationship is defined between the special axes.

Path axes

Path axes are interpolated together (all the path axes of a channel have a common path interpolator).

All the path axes of one channel have the same acceleration phase, constant travel phase and delay phase.

Positioning axes

Positioning axes are interpolated separately (each positioning axis has its own axis interpolator). Each positioning axis has its own feedrate and acceleration characteristic.

Synchronized axes

Synchronous axes are interpolated together with path axes (all path axes and synchronous axes of one channel have a common path interpolator).

All path axes and all synchronous axes of a channel have the same acceleration phase, constant travel phase and deceleration phase.

Axis configuration

The machine data below are used to assign the geometry axes, special axes, channel axes and machine axes as well as the names of the individual axis types:

MD20050 $MC_AXCONF_GEOAX_ASIGN_TAB (assignment of geometry axis to channel axis)

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

MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in channel)

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

MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX (assignment of spindle to machine axis)

Replaceable geometry axes

The "Replaceable geometry axes" function allows the geometry axes in a grouping to be replaced by other channel axes.

Axes that are initially configured as synchronous special axes in a channel can replace any selected geometry axis in response to a program command.

Link axis

Link axes are axes, which are physically connected to another NCU and whose position is controlled from this NCU. Link axes can be assigned dynamically to channels of **another** NCU. Link axes are not local axes from the perspective of a particular NCU.

The **axis container** concept is used for the dynamic modification of the assignment to an NCU. Axis replacement with `GET` and `RELEASE` from the part program is not available for link axes across NCU boundaries.

The link axes are described in
References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

Axis container

An axis container is a circular buffer data structure, in which local axes and/or link axes are assigned to channels. The entries in the circular buffer can be **shifted cyclically**.

In addition to the direct reference to local axes or link axes, the link axis configuration in the logical machine axis image also allows references to axis containers.

This type of reference consists of:

- Axis container number
- A slot (circular buffer location within the corresponding container)

The entry in a circular buffer location contains:

- A local axis

  or

- A link axis

The axis container function is described in
References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

9.1.2 Coordinate systems

MCS

The machine coordinate system (MCS) has the following properties:

- It is defined by the machine axes.
- The machine axes can be perpendicular to each other to form Cartesian system or arranged in any other way.
- The names of the machine axes can be defined.
- The machine axes can be linear or rotary axes.

BCS

The basic coordinates system (BKS) has the following properties:

- The geometry axes form a perpendicular Cartesian coordinate system.
- The BCS is derived from a kinematic transformation of the MCS.

BZS

The basic zero system (BZS) is the basic coordinate system with a basic offset.

SZS

The settable zero system (SZS) is the workpiece coordinate system with a programmable frame from the viewpoint of the WCS. The workpiece zero is defined by the settable frames `G54` to `G599`.

WCS

The workpiece coordinate system (WCS) has the following properties:

- In the workpiece coordinate system all the axes coordinates are programmed (parts program).
- It is made up of geometry axes and special axes.
- Geometry axes always form a perpendicular Cartesian coordinate system
- Special axes form a coordinate system without any geometrical relation between the special axes.
- The names of the geometry axes and special axes can be defined.
- The workpiece coordinate system can be translated, rotated, scaled or mirrored with FRAMES (TRANS, ROT, SCALE, MIRROR).

  Multiple translations, rotational movements, etc., are also possible.

Zero offset external

The zero offset external has the following properties:

- At a time defined in the PLC, a predefined additional zero offset between the basic and the workpiece coordinate systems is activated.
- The magnitudes of the offsets can be set by the following for each of the axes involved:
  - PLC
  - Operator Panel
  - Part program
- Activated offsets take effect at the instant the first motion block of the relevant axes is processed after offset activation. The offsets are superimposed on the programmed path (no interpolation).

  The velocity, at which the zero offset external is applied, is as follows:

  Programmed F value plus +1/2 JOG velocity

  The zero offset external is traversed at the end of G0 blocks.
- The activated offsets are retained after RESET and end of program.
- After POWER ON, the last active offset is still stored in the control but must be reactivated by the PLC.

9.1.3 Frames

Frame

A frame is a closed calculation rule (algorithm) that translates one Cartesian coordinate system into another.

Frame components

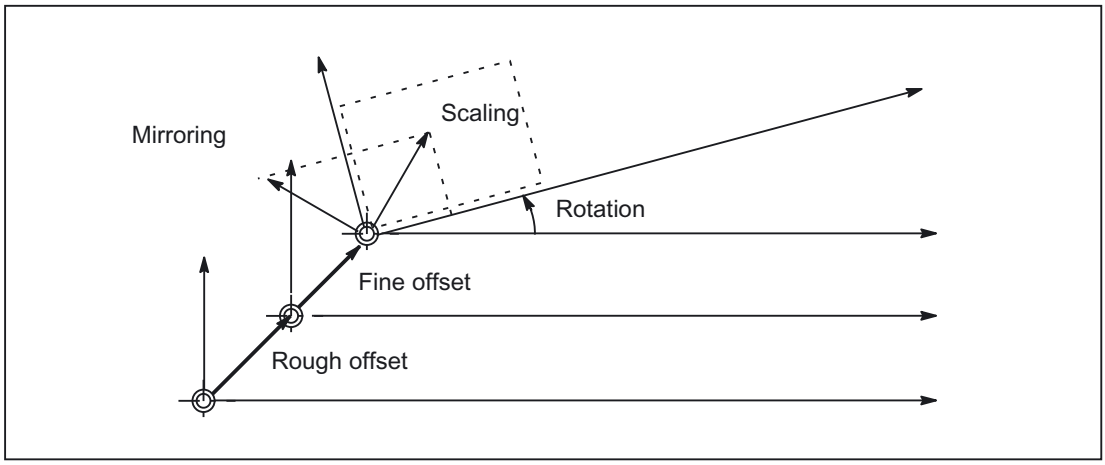


Figure 9-1 Frame components

A frame consists of the following components:

Frame components		Programmable with:
Offset	Coarse offset	TRANS ATRANS (additive translation component) CTRANS (zero offset for multiple axes) G58 (axial zero offset)
	Fine offset	CFINE G59 (axial zero offset)
Rotation		ROT / ROTS AROT / AROTS CROTS
Scaling		SCALE ASCALE
Mirroring		MIRROR AMIRROR

Features in relation to axes

The coarse and fine offsets, scaling and mirroring can be programmed for geometry and special axes. A rotation can also be programmed for geometry axes.

Coarse and fine offsets

The translation component of frames comprises:

- Coarse offset with TRANS, ATRANS and CTRANS

  The coarse offset is normally specified by the machine setter.

  The programmable offsets for all geometry axes and special axes are specified with TRANS.

- Fine offset with CFINE

  This can be defined by the machine operator, within certain input limits.

G58, G59 (only 840D sl)

For SINUMERIK 840 D sl,G58 and G59 can be programmed to replace the coarse and fine offsets of the programmable frame on an axial basis. These functions can only be used when the fine offset is configured.

- Coarse offset with G58

  G58 changes only the absolute translation component (coarse offset) for the specified axis; the total of additively programmed translations (fine offset) is retained.

- Fine offset with G59

  G59 is used for axial overwriting of the additively programmed translations for the specified axes that were programmed with ATRANS.

Frame rotations

Orientations in space are defined via frame rotations as follows:

- Rotation with ROT defines the individual rotations for all geometry axes.
- Solid angles with ROTS, AROTS, CROTS define the orientation of a plane in space.
- Frame rotation with TOFRAME defines a frame with a Z axis pointing in the tool direction.

Scaling

SCALE is used to program the programmable scale factors for all geometry axes and special axes.

If a new scaling is to be based on a previous scaling, rotation, translation or mirroring, then ASCALE must be programmed.

Mirroring

The axis to be mirrored can be set via the following machine data:

MD10610 MIRROR_REF_AX (reference axis for the mirroring)

Value	Meaning
0	Mirroring is performed around the programmed axis.
1, 2 or 3	Depending on the input value, mirroring is mapped onto the mirroring of a specific reference axis and rotation of two other geometry axes.

Frame chaining

Frame components or complete frames can be combined using the concatenation operator ":" to create a complete frame. For instance, the actual frame `$P_ACTFRAME` comprises chaining the complete basic frame, adjustable frame, the systems frames and the programmable frame:

$P_ACTFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
$P_ACTBFRAME : $P_IFRAME : $P_TOOLFRAME :
$P_WPFRAME : $P_TRAFRAME: $P_PFRAME
$P_ISO4FRAME : $P_CYCFRAME

Frames with G91

Incremental programming with `G91` is defined such that the compensation value is traversed additively to the incrementally programmed value when a zero offset is selected.

The behavior depends on the setting in the setting data:

SD42440 $SC_FRAME_OFFSET_INCR_PROG (zero offset in frames)

Value	Meaning
1	Zero offset is applied on `FRAME` and incremental programming of an axis (= default setting).
0	Only the programmed path is traversed.

Suppression of frames

The actual frames can be suppressed with the following operations:

Command	Meaning
G53	Suppression of the actual zero offset (non-modal)
G153	Suppression of the actual frame including basic frame
SUPA	Suppression of actual zero offset, including programmed offsets, system frames, handwheel offsets (DRF), external zero offset and overlaid motion

NCU global basic frames

Using NCU global basic frames, frames for other channels can be **pre-assigned** from a channel.

Properties of the NCU global basic frames:

- Can be read and written from all channels
- Can be activated only in the channels
- Offsets, scaling and mirroring for channel and machine axes

All global and channel-specific basic frames are chained and therefore a complete basic frame is obtained. As standard, there is at least one basic frame available per channel.

Settable frames

Adjustable frames can be defined as either global NCU or channel-specific frames.

Consistency

When writing, reading and activating frames, e.g. using channel coordination, the user is solely responsible for achieving consistent behavior within the channels. Cross-channel activation of frames is not supported.

9.2 Axes

9.2.1 Overview

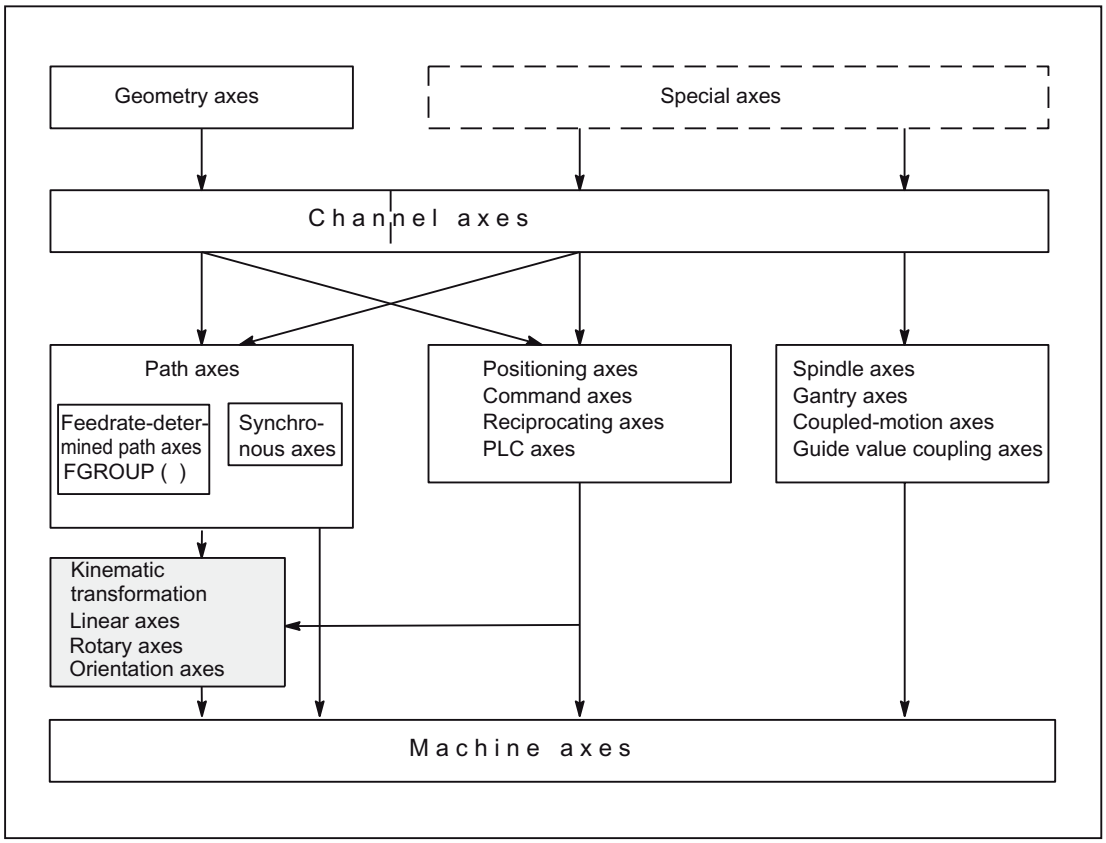


Figure 9-2 Relationship between geometry axes, special axes and machine axes

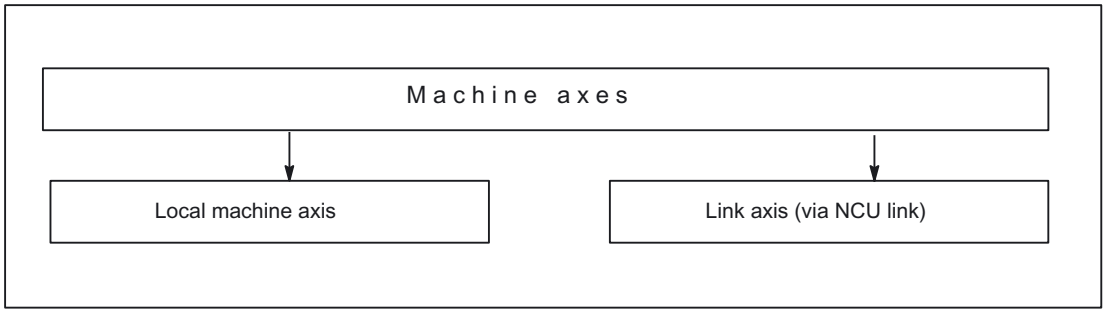


Figure 9-3 Local and external machine axes (link axes)

9.2.2 Machine axes

Meaning

Machine axes are the axes that actually exist on a machine tool.

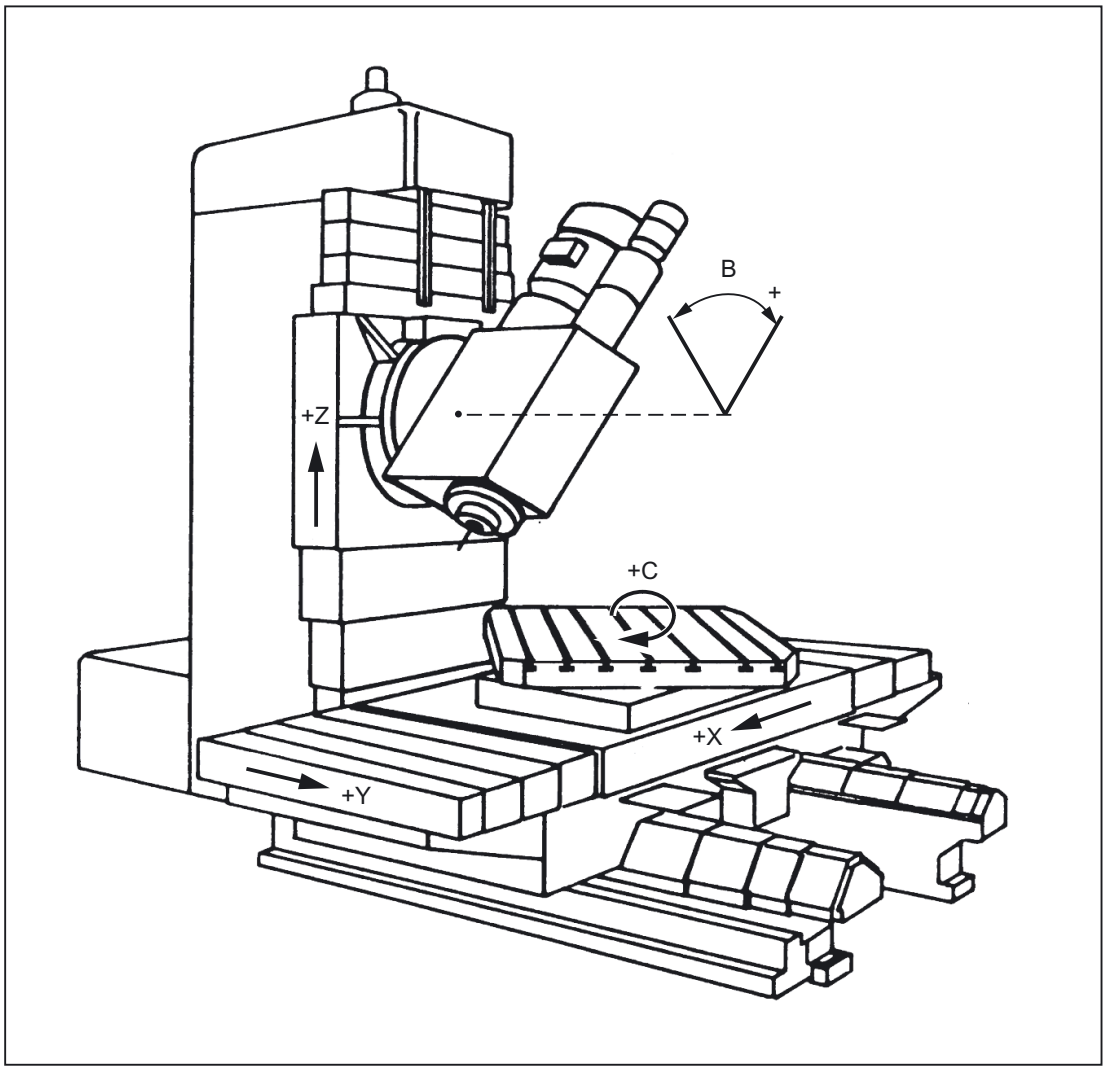


Figure 9-4 Machine axes X, Y, Z, B, S on a Cartesian machine

Application

The following can be machine axes:

- Geometry axes X, Y, Z
- Orientation axes A, B, C
- Loader axes
- Tool turrets
- Axes for tool magazine
- Axes for automatic tool changer
- Spindle sleeves
- Axes for pallet changers
- Etc.

9.2.3 Channel axes

Meaning

Each geometry axis and each special axis is assigned to a channel. Geometry axes and additional axes are always traversed in "their" channel.

9.2.4 Geometry axes

Meaning

The three geometry axes always make up a fictitious rectangular coordinate system.

By using FRAMES (offset, rotation, scaling, mirroring), it is possible to image geometry axes of the workpiece coordinate system (WCS) on the BCS.

Application

Geometry axes are used to program the workpiece geometry (the contour).

Plane selection `G17`, `G18` and `G19` (DIN 66217) always refers to the three geometry axes. That is why it is advantageous to name the three geometry axes X, Y and Z.

9.2.5 Replaceable geometry axes

Meaning

The "Replaceable geometry axes" function allows the geometry axes in a grouping to be replaced by other channel axes.

Axes that are initially configured as synchronous special axes in a channel can replace any selected geometry axis in response to a program command.

Example

On a machine with two Z axes, Z1 and Z2, either of the Z axes can be programmed as the geometry axis in response to an instruction in the part program.

Activation

Axis replacement is activated by the program command:

```
GEOAX([n, channel axis name]...)
```

n=0:	Removes an axis from the geometry axis grouping.
n=1, 2, 3:	Index of the geometry axis
GEOAX( ):	Establishes the basic setting defined via MD for the assignment of channel axes to geometry axes.
Channel axis name:	Name of channel axis, which is to operate as a geometry axis.

A channel axis, which has been designated a geometry axis, can only be addressed under its geometry axis name. The geometry axes names themselves remain unchanged.

Geometry axes can be replaced either individually or as a group in one command.

Supplementary conditions

As a basic rule, any channel axis designated as a geometry axis can be replaced by another channel axis.

In this case, the following restrictions apply:

- Rotary axes may not be programmed as geometry axes.
- A geometry axis, which has the same name as a channel axis, cannot be replaced by another channel axis (alarm message). Nor can an axis of this type be removed from the geometry axis grouping. It cannot change its position within the geometry axis grouping.
- Both axes in each of the axis pairs involved in the replacement operation must be block-synchronized.
- The following functions may not be active when geometry axes are replaced:
  - Transformation
  - Spline interpolation
  - Tool radius compensation
  - Tool fine compensation
- Any active DRF offset or zero offset external will remain operative. They both act on channel axes. The channel axis assignment is not affected by the replacement of geometry axes.

Replacement of geometry axes

All frames, protection zones and working area limitations are deleted. They may need to be reprogrammed after the replacement operation.

The system response to replacement of geometry axes is, therefore, identical to its response to a change (switch on/off, switchover) in a kinematic transformation.

Tool length compensation

Any active tool length compensation remains operative and is applied to the new geometry axes after replacement.

The system treats tool length compensations as not yet applied for the following geometry axes:

- All geometry axes, which have been newly added to the geometry axis grouping
- All geometry axes, which have changed their positioning within the geometry axis grouping

Geometry axes, which retain their position within the geometry axis grouping after a replacement operation, also retain their status with respect to tool length compensation.

RESET

The reset behavior of the changed geometry axis assignment is defined with the following machine data:

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

MD20118 $MC_GEOAX_CHANGE_RESET (allow automatic geometry axis change)

MD20110 $MC_RESET_MODE_MASK

Bit	Value	Meaning
12	0	In case of set machine data MD20118 $MC_GEOAX_CHANGE_RESET (allow automatic geometry axis change) a changed geometry axis assignment is deleted during reset or part program end. The initial setting defined in the machine data for the geometry axis assignment becomes active.
	1	A modified geometry axis assignment remains active after a reset/part-program end.

MD20118 $MC_GEOAX_CHANGE_RESET

Value	Meaning
0	The current configuration of the geometry axes remains unchanged on reset and program start. With this setting, the response is identical to older software versions without geometry axis replacement.
1	The configuration of the geometry axis remains unchanged during reset or part program end as a function of machine data MD20110 $MC_RESET_MODE_MASK and during part program start as a function of machine data MD20112 $MC_START_MODE_MASK (definition of initial control system settings with NC START) or brought to the initial status defined in the machine data MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB (assignment of geometry axis to channel axis).

Program start

Analogously to the Reset response, the behavior during program start is based on the setting in the machine data:

MD20112 $MC_START_MODE_MASK (definition of initial control system settings at NC-START)

Bit	Value	Meaning
12	0	A modified geometry axis assignment remains active on part-program start.
	1	A modified geometry axis assignment is cleared on part-program start.

Approaching a reference point

When the "Reference point approach" mode is selected, the geometry axis configuration defined by the machine data is automatically set.

M code

A changeover of the geometry axis with `GEOAX( )` can be communicated to the PLC through the output of an M code:

MD22532 $MC_GEOAX_CHANGE_M_CODE (M code at toolholder change)

Note

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

Transformation changeover

The following interrelationships must be noted with respect to kinematic transformation and geometry axis replacement:

- Geometry axis assignments cannot be modified when the transformation is active.
- Activation of a transformation deletes the programmed geometry axis configuration and replaces it by the geometry axis assignment stored in the machine data of the activated transformation.
- The initial setting defined through MD for the geometry axis configuration becomes effective after deactivating the transformation.

Should it be necessary to modify the geometry axis assignment in connection with transformations, then another new transformation must be configured. The total number of the transformations simultaneously available in the channel is equal to 8.

A maximum of two transformations per channel can be available simultaneously from the transformation groups below:

- Orientation transformations

  (3-axis, 4-axis, 5-axis and nutation transformation)
- `TRAANG` (oblique axis)
- `TRANSMIT`
- `TRACYL`

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

Function Manual, Extended Functions; Kinematic Transformation (M1)

Example

In the example below, it is assumed that there are 6 channel axes with channel axis names XX, YY, ZZ, U, V, W and three geometry axes with names X, Y, Z. The basic setting is defined in machine data so that the geometry axes are mapped on the first three channel axes, i.e. on XX, YY and ZZ.

Program code	Comment
GEOAX()	; The geometry axis assignment defined via the machine data MD AXCONF_GEOAX_ASSIGN_TAB is effective, i.e. XX, YY and ZZ become geometry axes.
G0 X0 Y0 Z0 U0 V0 W0	; All the axes in rapid traverse to position 0.
GEOAX (1, U, 2, V, 3, W)	; Channel axis U becomes the first, V the second and W the third geometry axis.
GEOAX(1, XX, 3, ZZ)	; Channel axis XX becomes the first, ZZ the third geometry axis. The second geometry axis remains unchanged.
G17 G2 X20 I10 F1000	; Semicircle in the X, Y plane. Channel axes XX and V traverse.
GEOAX(2,W)	; Channel axis W becomes the second geometry axis. The first and third geometry axes remain unchanged.
G17 G2 X20 I10 F1000	; Full circle in the X, Y plane. Channel axes XX and W traverse.
GEOAX()	; The geometry axis assignment defined via the machine data MD AXCONF_GEOAX_ASSIGN_TAB is effective, i.e. XX, YY and ZZ become geometry axes.
GEOAX (1, U, 2, V, 3, W)	; U, V and W become the first, second and third geometry axes.
G1 X10 Y10 Z10 XX=25	; Channel axes U, V, W each traverse to position 10, XX traverses to position 25.
GEOAX(0,V)	; V is again removed from the geometry axis grouping. U and W remain geometry axes. The second geometry axis is no longer assigned.
GEOAX (1, U, 2, V, 3, W)	; U, V and W become the first, second and third geometry axes, i.e. U and W remain unchanged.
GEOAX(3,V)	; V becomes the third geometry axis. This means that W, which was previously the third geometry axis, is removed from the geometry axis grouping. The second geometry axis is no longer assigned.

9.2.6 Special axes

Significance

In contrast to geometry axes, no geometrical relationship is defined between the special axes.

Note

Geometry axes have an exactly defined relationship in the form of a rightangled coordinate system.

Special axes are part of the basic coordinate system (BCS). With FRAMES (translation, scaling, mirroring), special axes of the workpiece coordinate system can be mapped on the basic coordinate system.

Application

Typical special axes are:

- Rotary axes
- Machine tool axes
- Tool revolver axes
- Loader axes

9.2.7 Path axes

Meaning

Path axes are interpolated together (all the path axes of a channel have a common path interpolator).

All the path axes of one channel have the same acceleration phase, constant travel phase and delay phase.

The feedrate programmed under address `F` (path feedrate) applies to all the path axes programmed in a block, with the following exceptions:

- An axis has been programmed that has been defined as having no control over the path velocity with instruction `FGROUP`.
- Axes programmed with instructions `POS` or `POSA` have an individual feedrate setting (axis interpolator).

Application

Path axes are used to machine the workpiece with the programmed contour.

9.2.8 Positioning axes

Meaning

Positioning axes are interpolated separately (each positioning axis has its own axis interpolator). Each positioning axis has its own feedrate and acceleration characteristic. Positioning axes can be programmed in addition to path axes (even in the same block). Path axis interpolation (path interpolator) is not affected by the positioning axes. Path axes and the individual positioning axes do not necessarily reach their block end points at the same time.

Instructions `POS` and `POSA` are used to program positioning axes and define block change criteria:

- `POS`

  Block change takes place when the path axes and positioning axes have reached their block end points.

- `POSA`

  Block change takes place when the path axes have reached their end of block position. Positioning axes continue to traverse beyond block limits to their block end point.

Concurrent positioning axes differ from positioning axes in that they:

- Only receive their block end points from the PLC
- Can be started at any time (not at block limits)
- Do not affect the execution of current part programs.

Application

Typical positioning axes are:

- Loaders for moving workpieces away from machine
- Tool magazine/turret

Reference

For further information, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)" and "S1: Spindles (Page 1383)".

References:

- Function Manual, Extended Functions; Positioning Axes (P2)
- Function Manual, Special Functions; Gantry Axes (G1)
- Function Manual, Special Functions; Axis Couplings and ESR (M3)
- Function Manual, Extended Functions; Oscillation (P5)
- Function Manual, Synchronized Actions

9.2.9 Main axes

Meaning

A main axis is an axis that is interpolated by the main run.

This interpolation can be started as follows:

- From synchronized actions

  (as command axes due to an event via block-related, modal or static synchronized actions)
- From the PLC via special function blocks in the basic PLC program

  (named as a concurrent positioning axis or a PLC axis)
- Via the setting data or from the part program

  (as an asynchronous or block-synchronous oscillating axis)

Channel control

An axis interpolated by the main axis reacts in terms of:

- NC stop
- Alarm handling
- Program control
- End of program
- RESET

Note

The response at the end of the program varies. The axis movement need not always be completed and, therefore, may carry on beyond the end of the program.

Application

Certain axes in the main run can be decoupled at the channel response triggered by the NC program sequence and controlled from the PLC. These axes are also interpolated in the main run and respond independently for the channel and program sequence.

A PLC-controlled axis can then be controlled independently by the NC. This concerns the following actions:

- The sequence for canceling the axis (equivalent to delete distancetogo)
- Stopping or interrupting the axis
- Continuing the axis (continue sequence of motion)
- Resetting the axis to its basic status

9.2.10 Synchronized axes

Meaning

Synchronous axes are components of the path axes, which are not referenced in order to calculate the tool path velocity. They are interpolated together with path axes (all path axes and synchronous axes of one channel have a common path interpolator).

All path axes and all synchronous axes of a channel have the same acceleration phase, constant travel phase and deceleration phase.

The feedrate (path feedrate) programmed under address F applies to all the path axes programmed in a block but not to the synchronous axes.

Synchronous axes take the same time to cover the programmed path as the path axes.

FGROUP command

The command FGROUP specifies whether the axis is a feed-defining **path axis** (used to calculate the path velocity) or a **synchronous axis** (not used to calculate the path velocity).

Example

Program code	Comment
N05 G00 G94 G90 M3 S1000 X0 Y0 Z0	;
N10 FGROUP(X,Y)	; Axes X/Y are path axes, Z is a synchronous axis.
N20 G01 X100 Y100 F1000	; Progr. feedrate 1000 mm/min. Feedrate of axis X = 707 mm/min. Feedrate of axis Y = 707 mm/min.
N30 FGROUP (X)	; Axis X is a path axis, axis Y is a synchronous axis
N20 X200 Y150	; Progr. Feedrate 1000 mm/min Feedrate of Axis X = 1000 mm/min Feedrate of Axis Y is set to 500 mm/min, because only half the distance is to be traversed.

Note

The channel axis name must be used for the FGROUP command.

This is defined by the machine data:

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

Application

In the case of helical interpolation FGROUP can be programmed to determine whether:

- The programmed feedrate should be valid on the path
  (all three programmed axes are path axes)
- The programmed feedrate should be valid on the circle
  (two axes are path axes and the infeed axis is a synchronous axis)

9.2.11 Axis configuration

Assigning geometry, special, channel and machine axes.

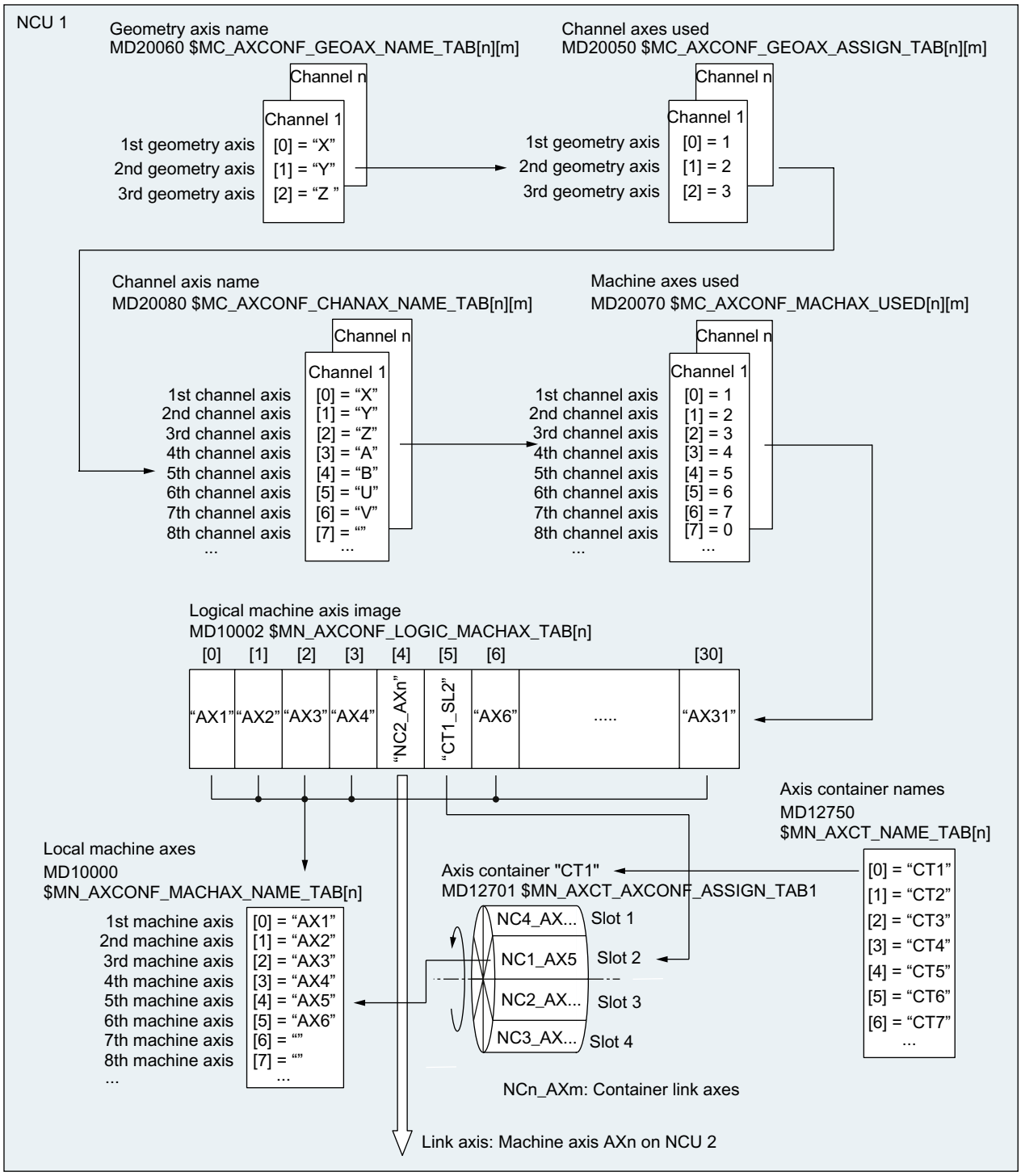


Figure 9-5 Axis configuration

Special features

- Leading zeros for user-defined axis identifiers are ignored:
  MD10000 `$MN_AXCONF_MACHAX_NAME_TAB[0] = X01 corresponds to X1
- The geometry axes must be assigned to the channel axes in ascending order without any gaps.
- All channel axes that are not geometry axes are special axes.

Channel axis gaps

Normally, each channel axis defined in machine data MD20080 $MC_AXCONF_CHANAX_NAME_TAB must be assigned a machine axis.

In order to simplify commissioning series of machines with a different number of machine axes, channel axes may also be defined, which are not assigned to any machine axis. As a result, gaps can occur in the numbering sequence of the channel axes.

Any channel axis gaps must be explicitly enabled:

MD11640 $MN_ENABLE_CHAN_AX_GAP = 1

Without being enabled, a value of 0 in machine data:

MD20070 $MC_AXCONF_MACHAX_USED

ends the assignment of possibly existing additional machine axes to channel axes.

References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

Note

Channel axes without assigned machine axes (channel axis gaps) are, regarding the number and indexing of the channel axes, treated just like normal channel axes with associated machine axes.

If a channel axis without assigned machine axis (channel axis gap) is defined as geometry axis, then this is rejected without an alarm.

Example: Channel axis gap

Channel axis B is not assigned a machine axis in the following example.

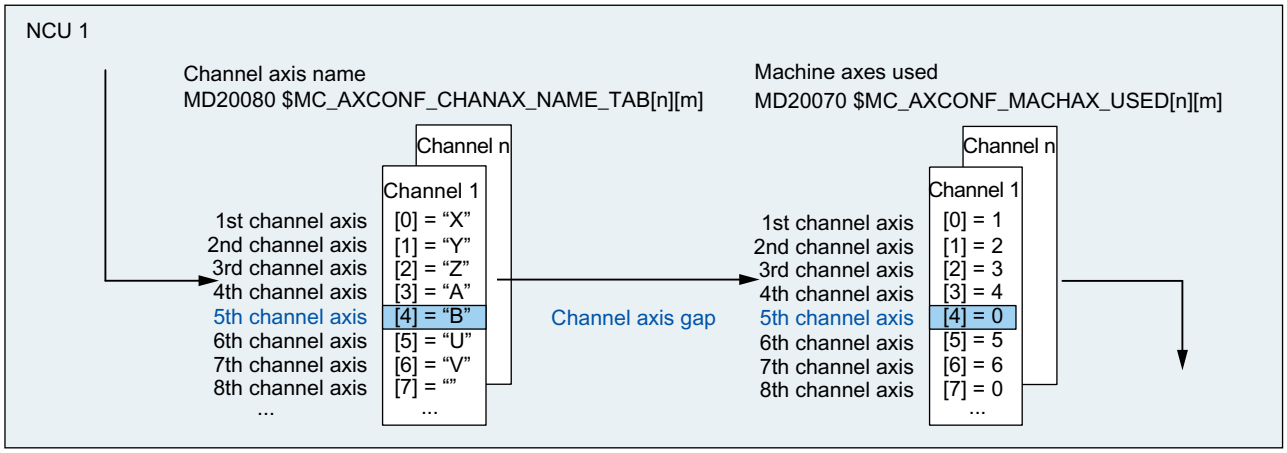


Figure 9-6 Axis configuration with channel axis gap (excerpt)

Special situations: Channel axis gaps

Regarding channel axis gaps, the following also have to be taken into account:

- Channel axes without assigned machine axes (channel axis gaps) are, regarding the number and indexing of the channel axes, treated just like normal channel axes with associated machine axes.
- If a channel axis without assigned machine axis (channel axis gap) is defined as geometry axis, then this is rejected without an alarm.

9.2.12 Link axes

Meaning

A link axis is a machine axis that is not on the NCU from which it is traversed. The identifier of a local machine axis is not entered in the machine data for the logical machine axis image of the traversing NCU, but the NCU and machine axis identifier of the NCU to which it is physically connected.

As an example, machine axis AX1 of NCU2 should be traversed from NCU1:

- NCU1: MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[n] = NC2_AX1

Requirement

The NCUs involved must be connected using link communication as a requirement for using link axes. The link axes and link communication functions are described in detail in:

References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

9.3 Zeros and reference points

9.3.1 Reference points in working space

Zeros and reference points

The neutral position of the machine is obtained from the coordinate axes and the constructive characteristics of the machine. The zero of the coordinate system is obtained by defining a suitable reference point on the machine in its neutral position.

The position of the coordinate systems (MCS, BCS, BZS, SZS, WCS) is determined by means of zeros.

Zero points		Reference points	
	**M** = Machine zero		**R** = Reference point
	**W** = Workpiece zero		**T** = Toolholder reference point

Machine zero M

The machine zero M defines the machine coordinate system MCS. All other reference points refer to the machine zero.

Workpiece zero W

The workpiece zero W defines the workpiece coordinate system in relation to the machine zero M. The programmed part-program blocks are executed in the workpiece coordinate system WCS.

Reference point R

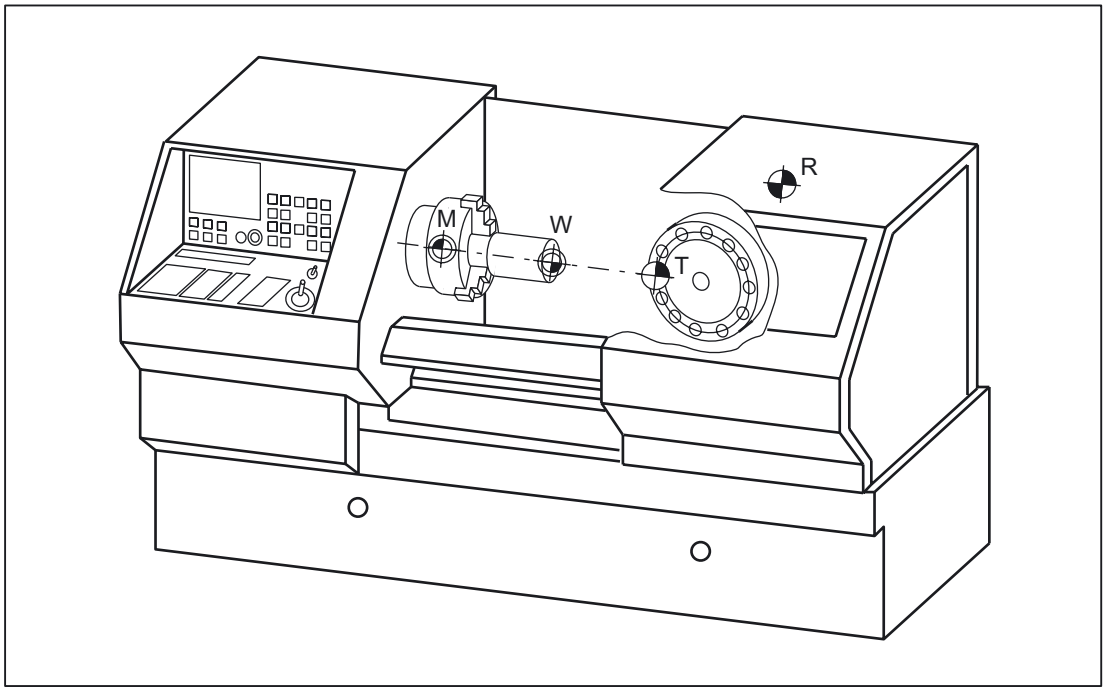
The position of the reference point R is defined by cam switches. Reference point R calibrates the position measuring system.

With incremental encoders, the reference point must be approached every time the control power is switched on. The control can only then work with the measuring system and transfer all position values to the coordinate systems.

Toolholder reference point T

The toolholder reference point T is located on the toolholder locator. By entering the tool lengths, the control calculates the distance between the tool tip (TCP Tool Center Position) and the toolholder reference point.

Example: Zeros and reference points on a turning machine



9.3.2 Position of coordinate systems and reference points

Control POWER ON

For incremental measuring probes, the reference point must be approached each time the control is activated so that the control can transfer all position values to the coordinate system.

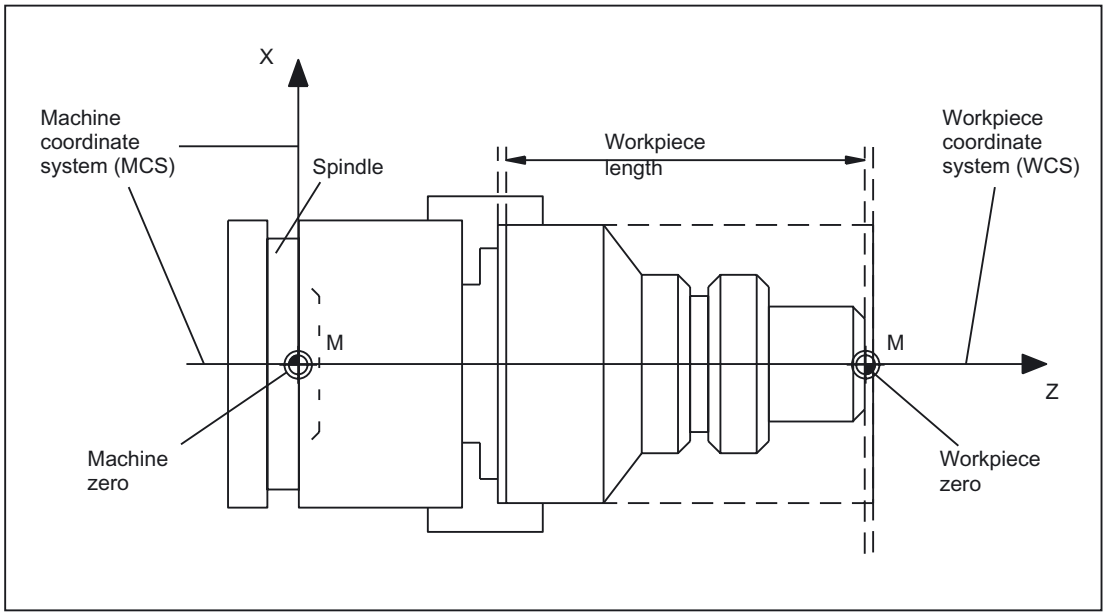


Figure 9-7 Position of coordinate systems by machine zero M and workpiece zero W

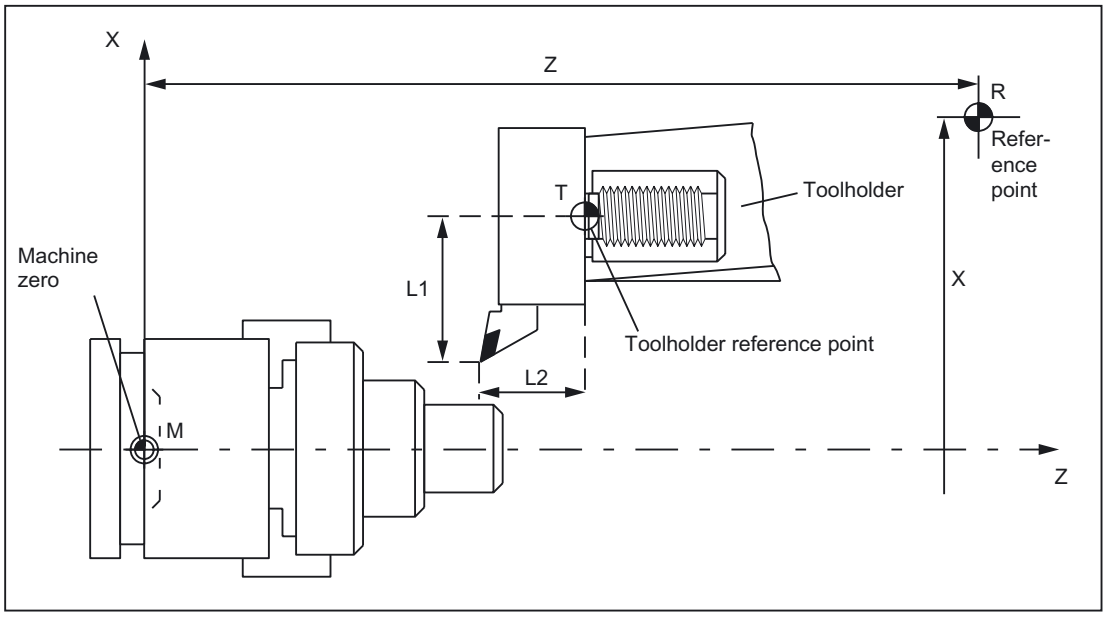


Figure 9-8 Position of reference point in relation to machine zero

9.4 Coordinate systems

9.4.1 Overview

Cartesian coordinate systems

DIN 66217 stipulates that machine tools must use right-angled, rectangular (Cartesian) coordinate systems. The positive directions of the coordainate axes are determined using the "Right Hand Rule". The coordinate system is related to the workpiece and programming takes place independently of whether the tool or the workpiece is being traversed. When programming, it is always assumed that the tool traverses relative to the coordinate system of the workpiece, which is intended to be stationary.

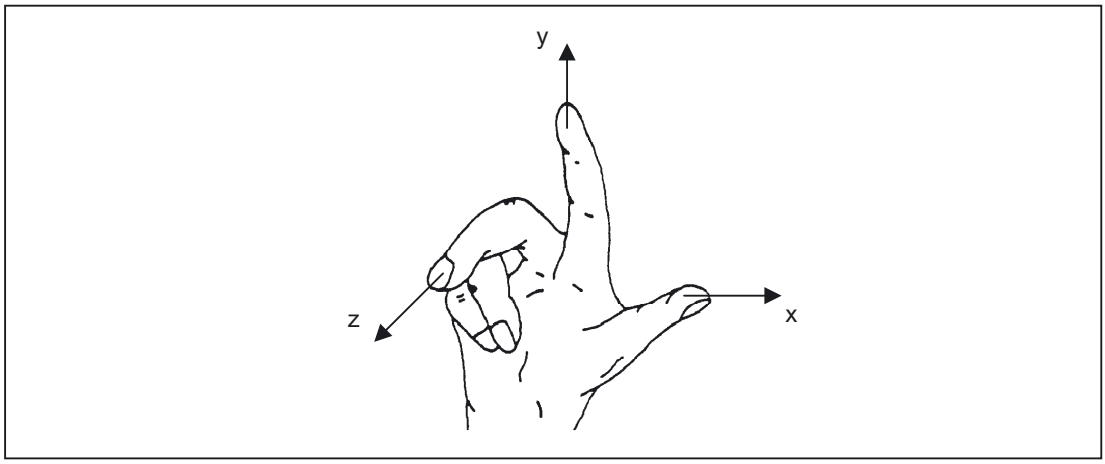


Figure 9-9 Right-hand rule

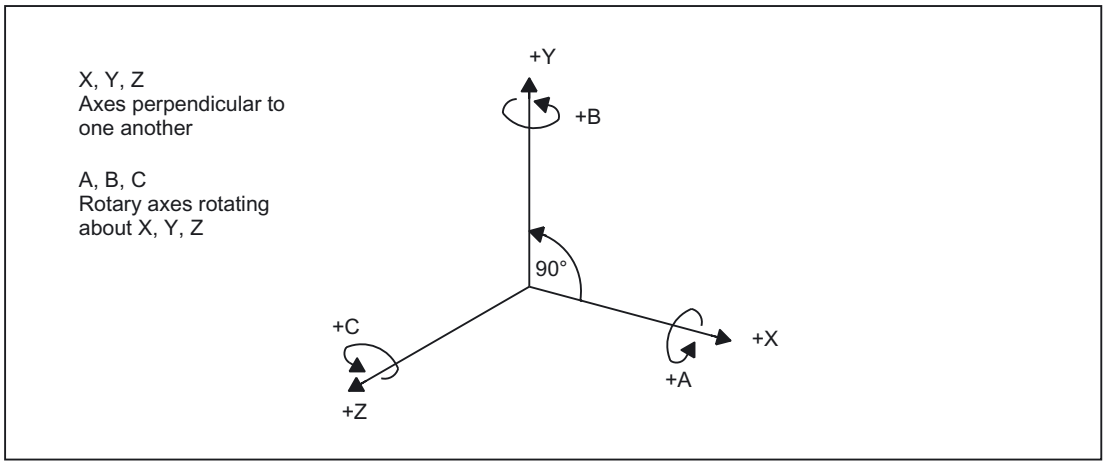


Figure 9-10 Clockwise, rectangular Cartesian coordinate system

The following coordinate systems are defined:

MCS	**M**achine **C**oordinat **S**ystem
BCS	**B**asic **C**oordinate **S**ystem
BZS	**B**asic **Z**ero **S**ystem
SZS	**S**ettable **Z**ero **S**ystem
WCS	**W**orkpiece **C**oordinate **S**ystem

Interrelationships between coordinate systems

The coordinate systems are determined by the kinematic transformation and the FRAMES.

A kinematic transformation is used to derive the BCS from the MCS. If no kinematic transformation is active, the BCS is the same as the MCS.

The basic frame maps the BCS onto the BKS.

An activated adjustable `FRAME G54...G599` ENS is derived from the BNS.

The WCS, which is the basis for programming, is defined by the programmable `FRAME`.

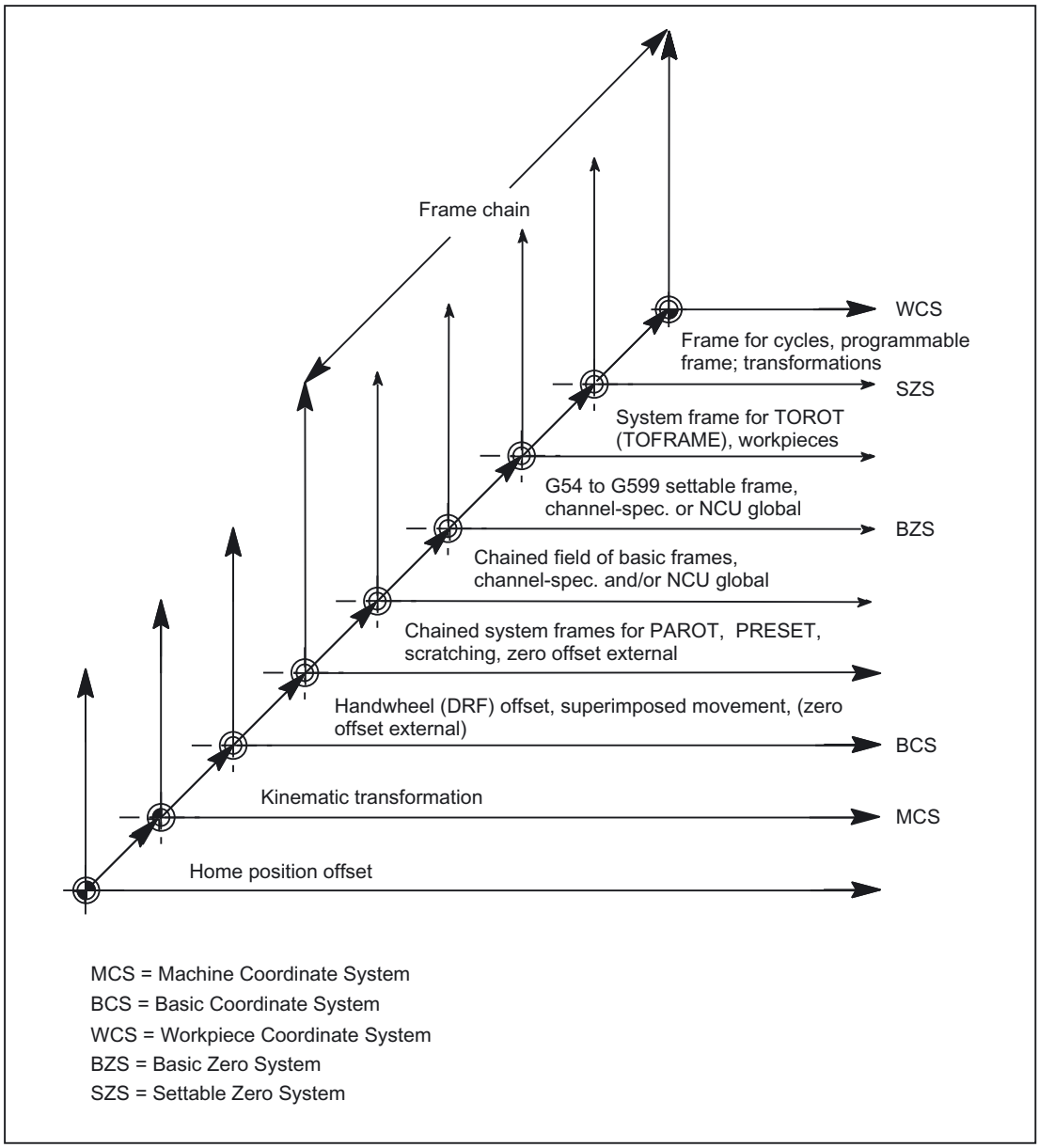


Figure 9-11 Interrelationships between coordinate systems

9.4.2 Machine coordinate system (MCS)

Machine coordinate system (MCS)

The machine coordinate system (MCS) is made up of all physically available machine axes.

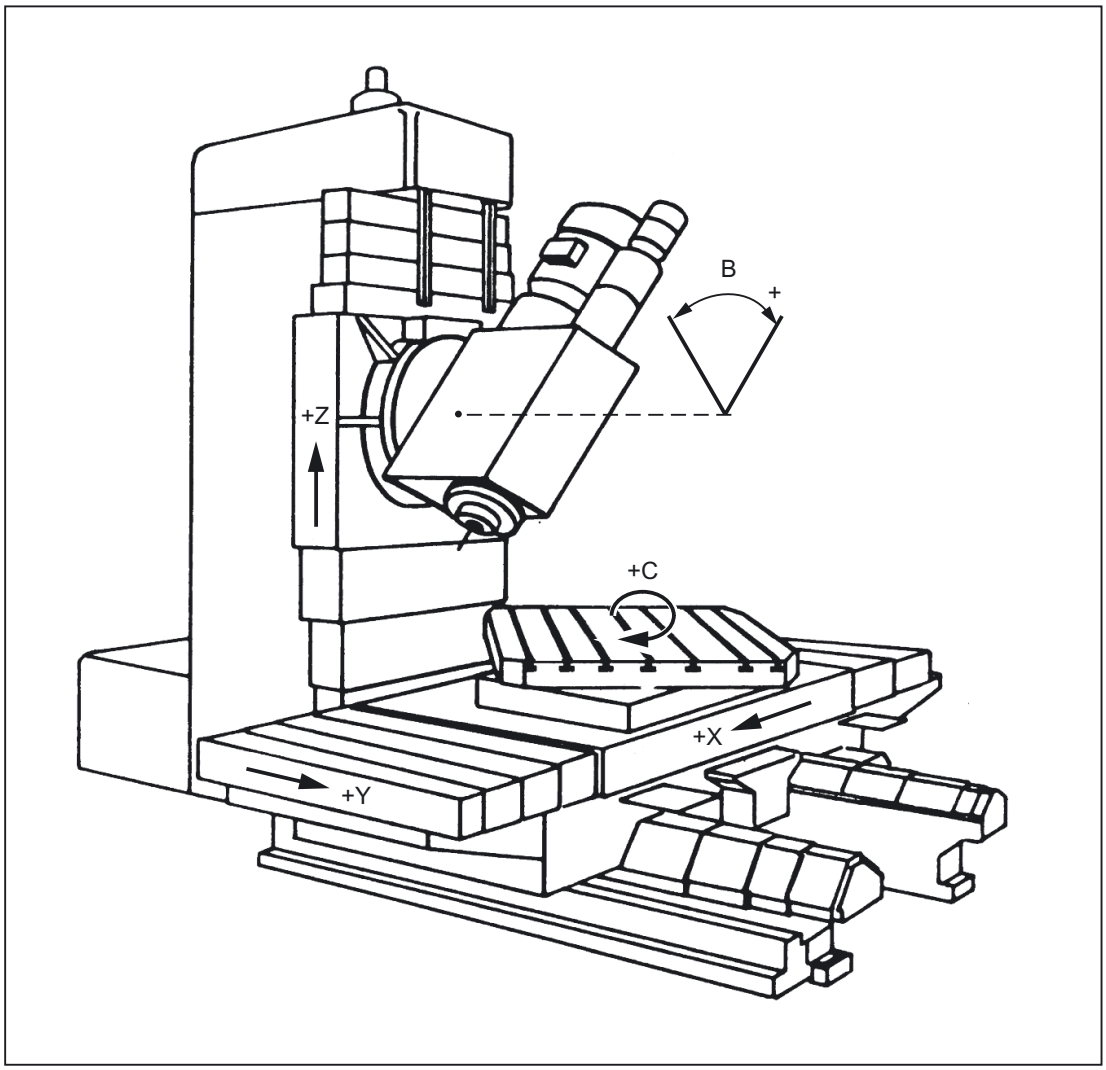


Figure 9-12 MCS with machine axes X, Y, Z, B, C (5-axis milling machine)

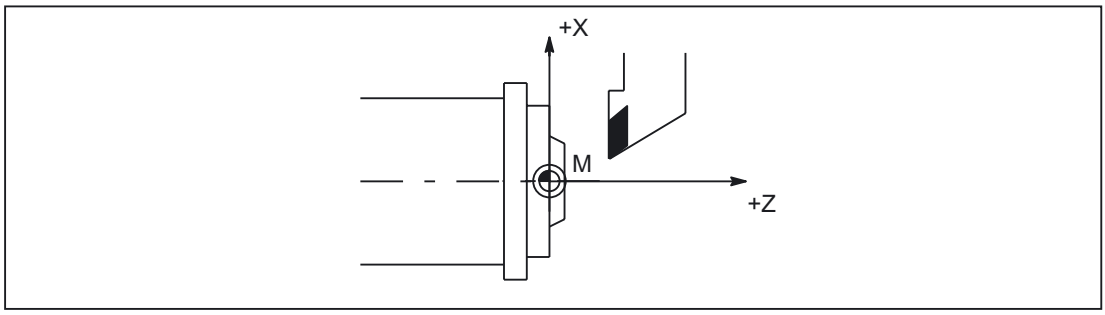


Figure 9-13 MCS with machine axes X, Z (turning machine)

Axial preset offset

The reference point of the control in the machine coordinate system (machine zero) can be reset via the "Preset offset (PRESETON)" function.

Note

We recommend that the function is **only** used for machine axes that do not require a reference point.

In order to restore the original machine coordinate system, the machine axis must be re-referenced, e.g. with G74 (reference point approach).

The machine axes are not moved with the preset offset.

⚠ CAUTION

After a preset offset, the appropriate machine axis is in the "Not referenced" state! This means that when using absolute encoders, the encoder adjustment is lost and must be performed again (e.g. by calibration with a laser interferometer). The use of PRESETON in combination with absolute encoders is therefore not recommended.

References

- Programming Manual, Fundamentals
  Section: "Supplementary commands" > "Reference point approach (G74)"
- Programming Manual, Job Planning
  Section: "Coordinate transformations (FRAMES)" > "Preset offset (PRESETON)

9.4.3 Basic coordinate system (BCS)

Basic coordinate system (BCS)

The basic coordinate system (BCS) consists of three mutually perpendicular axes (geometry axes) as well as other special axes, which are not interrelated geometrically.

Machine tools without kinematic transformation

BCS and MCS always coincide when the BCS can be mapped onto the MCS withouth kinematic transformation (e.g. TRANSMIT / face transformation, 5-axis transformation and up to three machine axes).

On such machines, machine axes and geometry axes can have the same names.

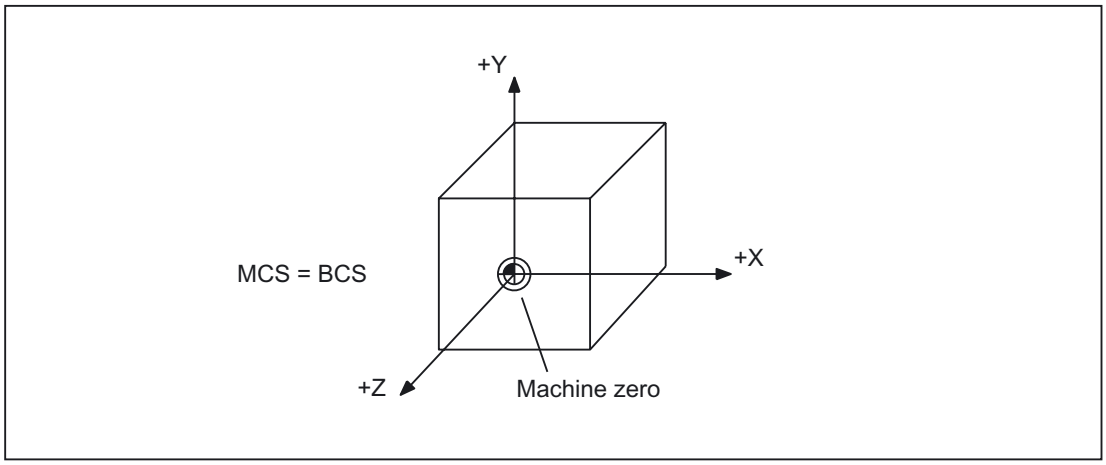


Figure 9-14 MCS=BCS without kinematic transformation

Machine tools with kinematic transformation

The BCS and MCS do not coincide when the BCS is mapped onto the MCS with kinematic transformation (e.g. TRANSMIT / face transformation, 5-axis transformation or more than three axes).

On such machines the machine axes and geometry axes must have different names.

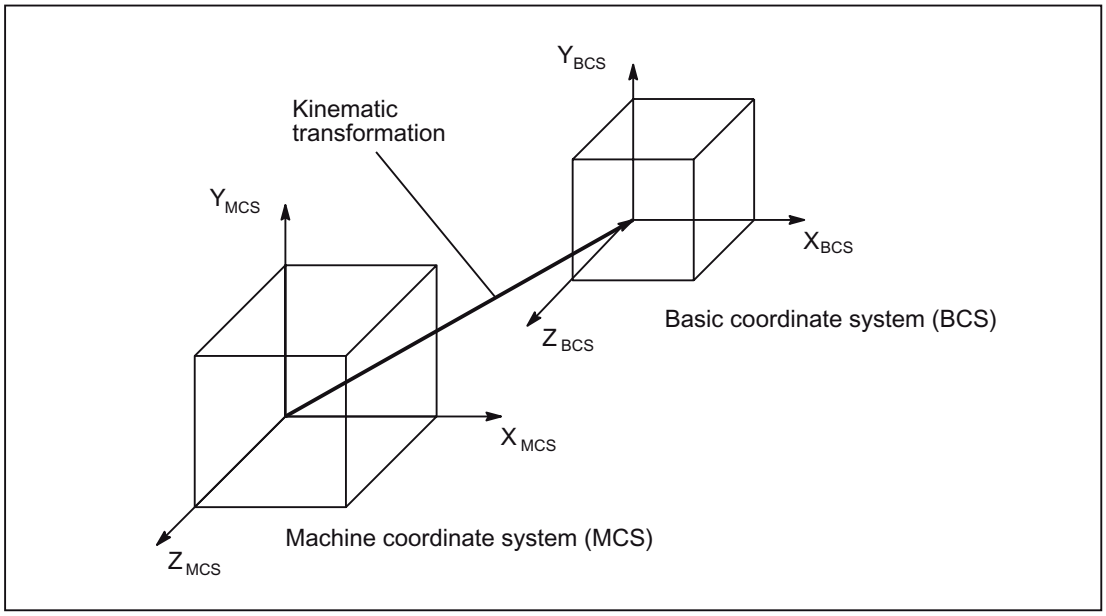


Figure 9-15 Kinematic transformation between the MCS and BCS

Machine kinematics

The workpiece is always programmed in a two- or three-dimensional, right-angled coordinate system (WCS). However, such workpieces are being programmed ever more frequently on machine tools with rotary axes or linear axes not perpendicular to one another. Kinematic transformation is used to represent coordinates programmed in the workpiece coordinate system (rectangular) in real machine movements.

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

Function Manual, Extended Functions; Kinematic Transformation (M1)

9.4.4 Additive offsets

External zero offsets

The "zero offset external" is an axial offset. Unlike with frames, no components for rotation, scaling and mirroring are possible.

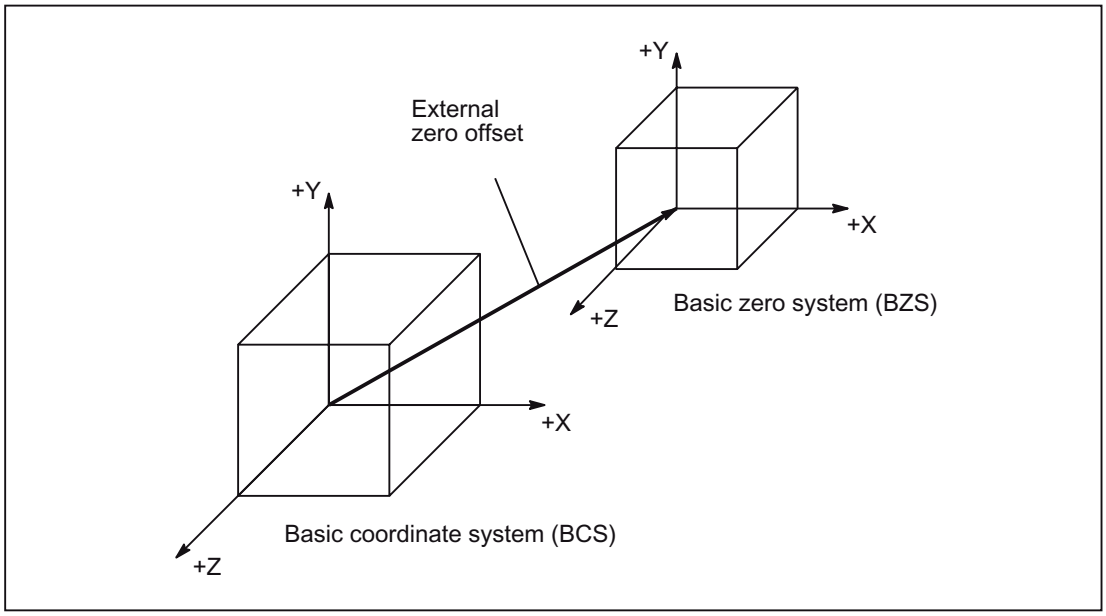


Figure 9-16 Zero offset external between BCS and BZS

Setting the offset values

The offset values are set:

- PLC

  By describing system variables
- Via the operator panel

  From menu "Current zero offsets"
- NC program

  By assigning to system variable $AA_ETRANS[axis]

Activation of the offset values

The 0/1 edge of the following PLC signal activates the previously defined offset values:

DB31, ... DBX3.0 (accept external zero offset)

The 0/1 edge change is only evaluated in Automatic operating mode.

Effect of activation

The offset for an axis becomes active when the first motion block for this axis is executed after the offset is activated.

Example of possible chronological sequence:

```
Program code          Comment
G0 X100
X150                  ; A new "Zero offset external" is activated by the PLC during
                        this motion.
X200                  ; The new "Zero offset external" is applied due to G0
                        programming at the end of the block (X200), if no velocity
                        reserve is available (100%).
```

The "Zero offset external via system frame" is applied immediately.

Channel-specific system frames can be configured through the following machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK (system frames SRAM)

Programming

Setting a new offset via the axis-specific system variables:

$AA_ETRANS[axis]=$R_i$

The instruction below reads the axis-specific active offset value:

R_i=$AA_ETRANS[axis]

Note

The read value can then differ from the previously set value, if the set value has not yet been activated.

The read value corresponds to a value set previously, if the most recently set value has not yet been activated. The system frame for the "Zero offset external" exists only if it has been configured.

DRF offset

The DRF offset enables the adjustment of an additional incremental zero offset for geometry and additional axes in the basic coordinate system **through handwheel**.

The DRF offset can be read via the axis-specific system variable:

$AC_DRF[<Axis>]

References:
Function Manual, Extended Functions; Manual and Handwheel Travel (H1),
Section: DRF offset

Overlaid movements

The "Superimposed motion" for the programmed axis can only be accessed from synchronized actions via the system variable $AA_OFF[axis].

Power-up

After run-up (POWER ON) the last used offset values for the "Zero offset external" are stored and do not become effective again until there is a renewed activation signal.

System frames are retained during Power ON, depending on the following machine data:

MD24008 $MC_CHSFRAME_POWERON_MASK (reset system frames after Power On)

RESET/end of program

The activated values remain active after RESET and program end.

Reset response of channel-specific system fames as follows:

The system frame for the "external zero offset" is active after RESET with the following machine data setting:

MD24006 $MC_CHSFRAME_RESET_MASK, bit 1 = 1

The "external zero offset" in the active system frame is deleted in the data management through the following machine data setting:

MD24006 $MC_CHSFRAME_RESET_MASK, bit 1 = 0

The following frames are active after RESET:

- System frame for:

  MD24006 $MC_CHSFRAME_RESET_MASK, Bit 4 = 1 (workpiece reference point)

  MD24006 $MC_CHSFRAME_RESET_MASK, Bit 5 = 1 (cycles)

Suppression

The NC program instruction SUPA suppresses the "Zero offset external" while the block is being processed.

The command G74 (reference point approach) and the equivalent operator actions in "Reference point approach" mode suppress the "Zero offset external" for the duration of the reference point approach.

With G74, i.e. "Automatic" or "MDA" mode, the previously active "External zero offset" automatically becomes active again with the next traversing motion in the block.

After a mode change from "Reference point approach" mode, the NC/PLC interface signal for the referenced axes must be set for reactivation.

9.4.5 Basic zero system (BZS)

Basic zero system (BZS)

The basic zero system (BZS) is the basic coordinate system with a basic offset.

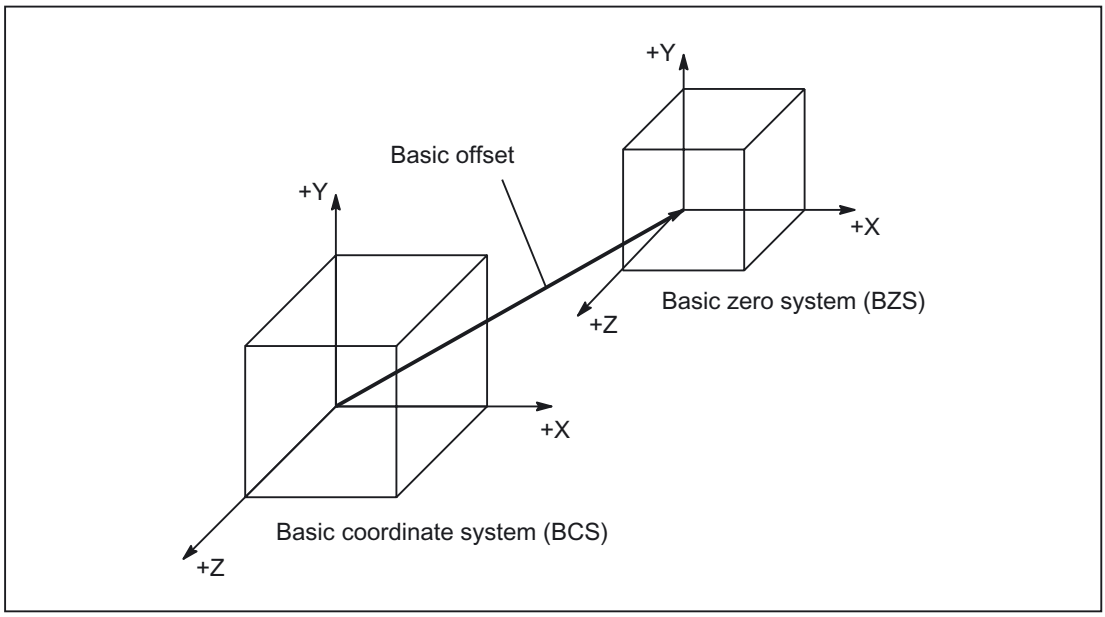


Figure 9-17 Basic offset between BCS and BZS

Basic offset

The basic offset describes the coordinate transformation between BCS and BZS. It can be used, for example, to define the palette window zero.

The basic offset comprises:

- Zero offset external
- DRF offset
- Superimposed motion
- Chained system frames
- Chained basic frames

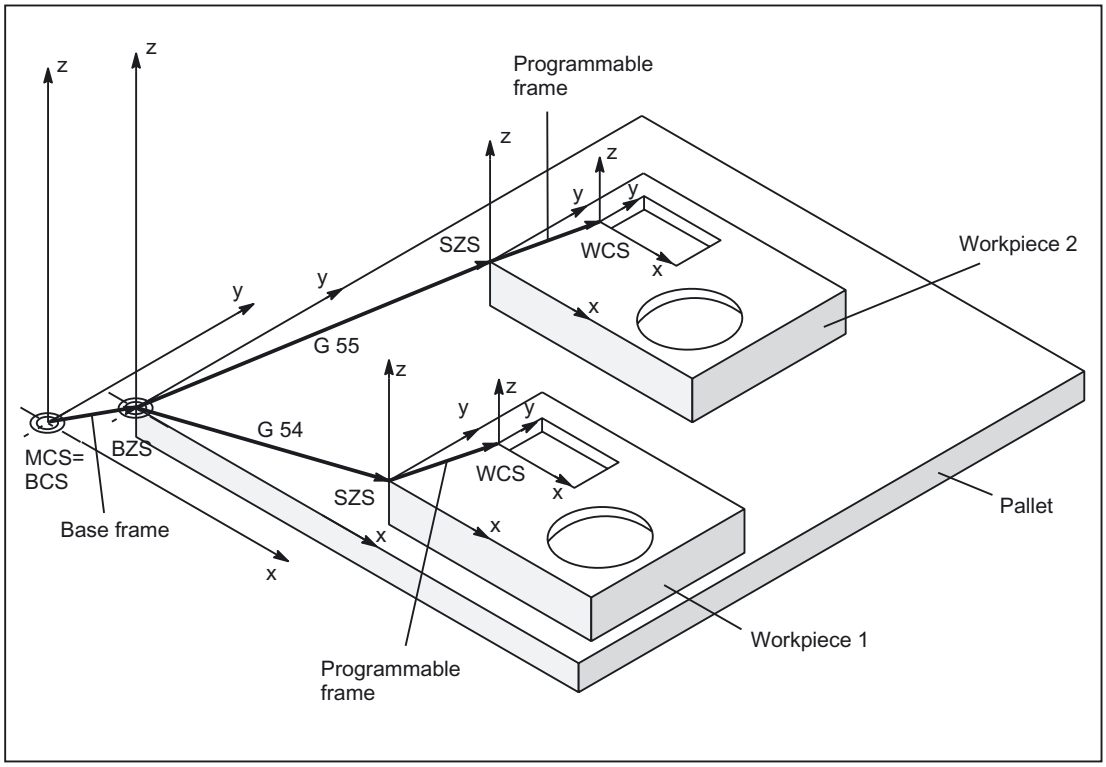


Figure 9-18 Example of the use of the basic offset

The following settings apply:

- The user can change the basic offset from the part program by means of an operator action and from the PLC.
- If the basic offset is to take effect immediately, an ASUB can be started via the PLC using FC9 in order to execute the appropriate G code.

Note

Recommendation to the machine manufacturer

Use the 3rd basic offset onwards for your own applications.

The 1st and 2nd basic offset are reserved for PRESET and the "Zero offset external".

9.4.6 Settable zero system (SZS)

Settable zero system (SZS)

The "settable zero system" (SZS) is the workpiece coordinate system WCS with a programmable frame (viewed from the perspective of the WCS). The workpiece zero is defined by the settable FRAMES G54 to G599.

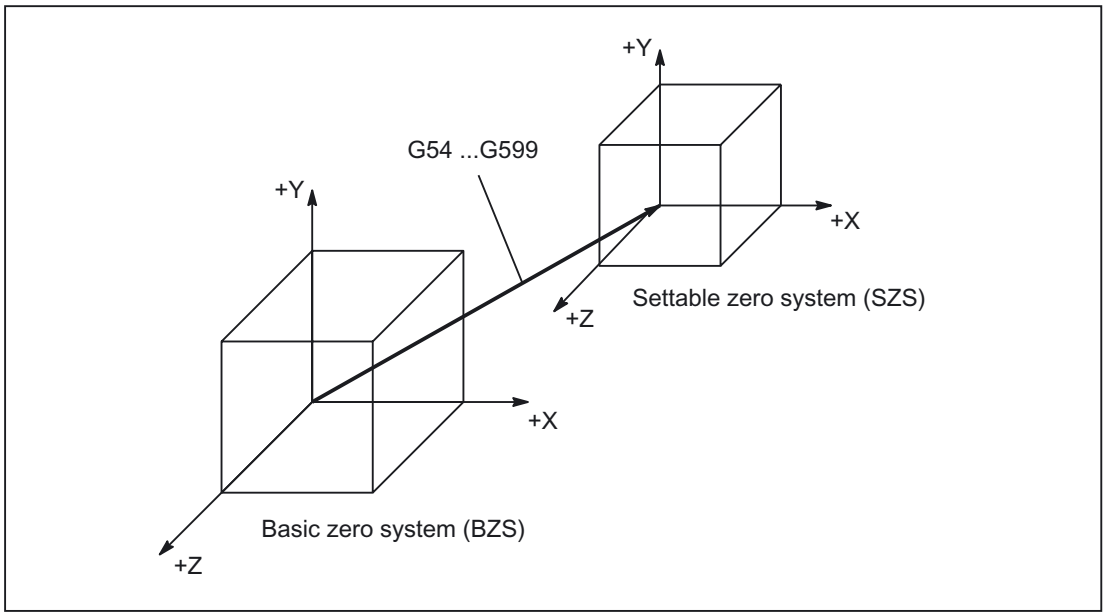


Figure 9-19 Settable FRAME G54 ... G599 between BNS and ENS

Programmable offsets act on the "settable zero system". All programmable offsets refer to the "settable zero system".

WCS actual-value display in WCS or SZS

The actual values of the axes in the machine coordinate system (MCS) or the WCS can be displayed on the HMI operator interface. For displays in WCS, the actual values can also be displayed in relation to the SZS. The corresponding parameterization takes place through the machine data:

MD9424 $MM_MA_COORDINATE_SYSTEM (coordinate system for actual value display)

Value	Significance
0	Actual-value display in relation to the WCS
1	Actual-value display in relation to the SZS

Note

Display of the current coordinate system

When "Actual-value display in relation to the SZS" is active, the WCS is still displayed on the HMI operator interface as the coordinate system to which the actual-value display relates.

Example

Actual-value display in relation to the WCS or SZS

Code (excerpt)	Actual value display: Axis X (WCS)	Actual value display: Axis X (SZS)
`N10 X100`	100	100
`N20 X0`	0	0
`N30 $P_PFRAME = CTRANS(X,10)`	0	10
`N40 X100`	100	110

9.4.7 Workpiece coordinate system (WCS)

Workpiece coordinate system (WCS)

The workpiece coordinate system (WCS) is the programming basis.

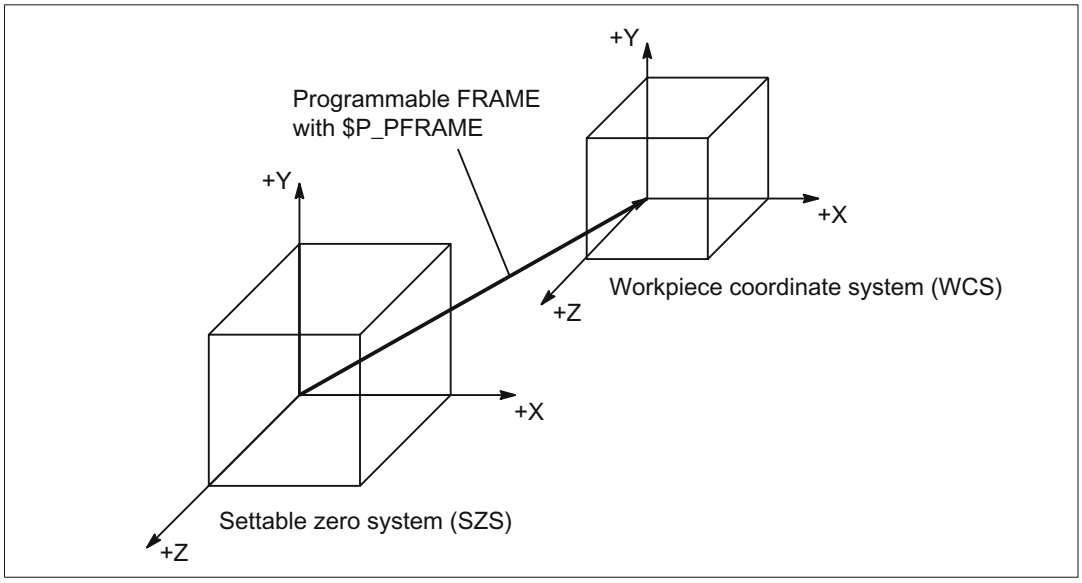


Figure 9-20 Programmable FRAME between SZS and WCS

9.5 Frames

9.5.1 Frame types

A frame is a data structure that contains values for offset (TRANS), fine offset (FINE), rotation (ROT), mirroring (MIRROR) and scaling (SCALE) for axes.

When activating the frame, using the frame values, a static coordinate transformation for the axes contained in the frame is performed using a defined algorithm.

Axial frame

An axial frame contains the frame values of an axis.

Example of the data structure of an axial frame for axis X:

Axis	TRANS	FINE	ROT	MIRROR	SCALE
X	10.0	0.1	0.0	0	1

Channel-specific frame

A channel-specific frame contains frame values for all channel axes (geometry, special and machine axes).

Rotations (ROT) are only included in the calculation for geometry axes.

A channel-specific frame is only active in the channel in which the frame is defined.

Example of the data structure of a channel-specific frame:

- Geometry axes: X, Y, Z
- Special axes: A
- Machine axes: AX1

Axis	TRANS	FINE	ROT	MIRROR	SCALE
X	10.0	0.1	0.0	0	1
Y	0.0	0.0	0.0	1	1
Z	0.0	0.0	45.0	0	1
A	2.0	0.1	0.0	0	2
AX1	0.0	0.0	0.0	0	0

Global frame

A global frame contains the frame values for all machine axes.

A global frame is active in all channels of the NC.

Example of the data structure of a channel-specific frame:

- Machine axes: AX1, ... AX5

Axis	TRANS	FINE	ROT	MIRROR	SCALE
AX1	10.0	0.1	-	0	1
AX2	0.0	0.0	-	1	1
AX3	0.0	0.0	-	0	1
AX4	2.0	0.1	-	0	2
AX5	0.0	0.0	-	1	1

9.5.2 Frame components

9.5.2.1 Translation

Programming

The programming of the translation or coarse offset can be performed via the following commands:

- Example of data management frames `$P_UIFR`
  - Complete frame: `$P_UIFR[<n>] = CTRANS(<K1>,<V1>[,<K2>,<V2>][,<K3>,<V3>])`
    with Km = coordinate x, y or z and Vm = offset m
  - Frame component: `$P_UIFR[<n>,<k>,TR] = <V>`
    with K = coordinate x, y or z and V = offset
- Example of programmable frame
  - TRANS `<K1> <V1> [<K2> <V2>][<K3> <V3>]`
    with Km = coordinate x, y or z and Vm = offset m

Programs examples:

Program code	Remark
`$P_UIFR[1] = CTRANS(X,10,Y,10)`	Complete frame
`$P_UIFR[1,X,TR] = 10`	Frame components
`TRANS X=10 Y=10`	Programmable frame

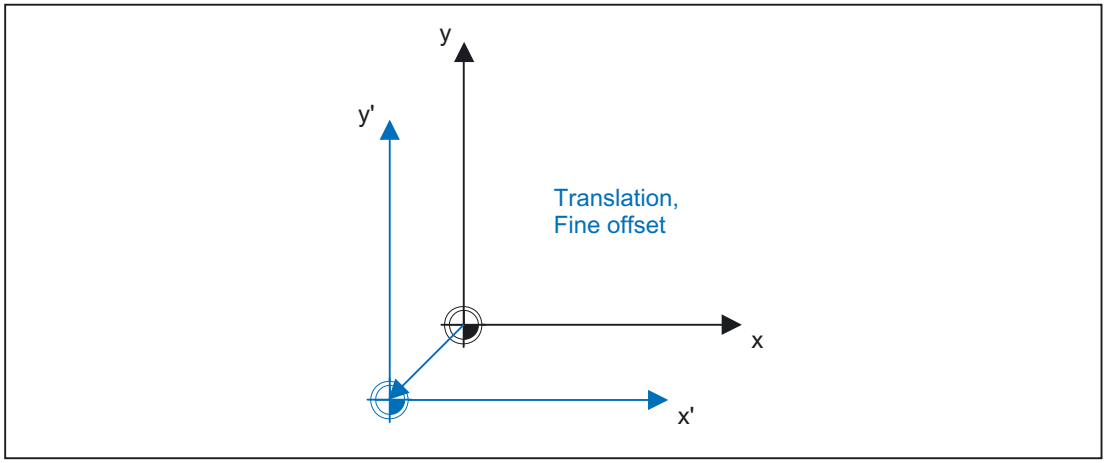


Figure 9-21 Offset in the Z direction

9.5.2.2 Fine offset

Parameterization

The fine offset is enabled via the machine data:

MD18600 $MN_MM_FRAME_FINE_TRANS = <value>

Value	Meaning
0	The fine offset cannot be entered or programmed.
1	Fine offset is possible for settable frames, basic frames and the programmable frame via command or program.

Programming

The programming of the translation or coarse offset can be performed via the following commands:

- Example of data management frames `$P_UIFR`
  - Complete frame: `$P_UIFR[<n>] = CFINE(<K1>,<V1>[,<K2>,<V2>][,<K3>,<V3>])`
    with Km = coordinate x, y or z and Vm = offset m
  - Frame component: `$P_UIFR[<n>,<K>,FI] = <V>`
    with K = coordinate x, y or z and V = offset
- Example of programmable frame
  - TRANS `<K1> <V1> [<K2> <V2>][<K3> <V3>]`
    with Km = coordinate x, y or z and Vm = offset m

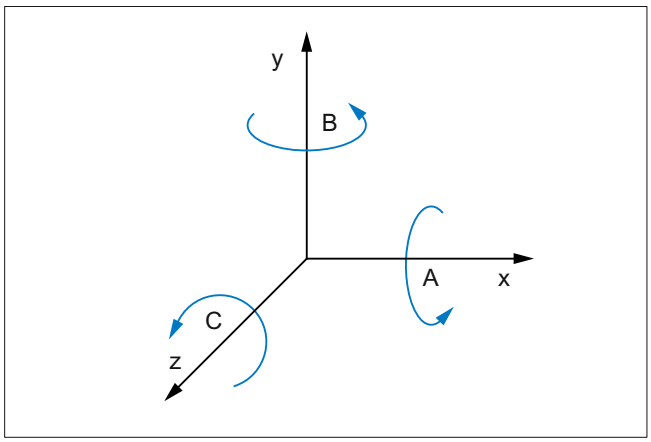
Programming examples:

Program code	Remark
`$P_UIFR[1] = CTRANS(X,10,Y,10)`	Complete frame
`$P_UIFR[1,X,TR] = 10`	Frame components
`TRANS X=10 Y=10`	Programmable frame

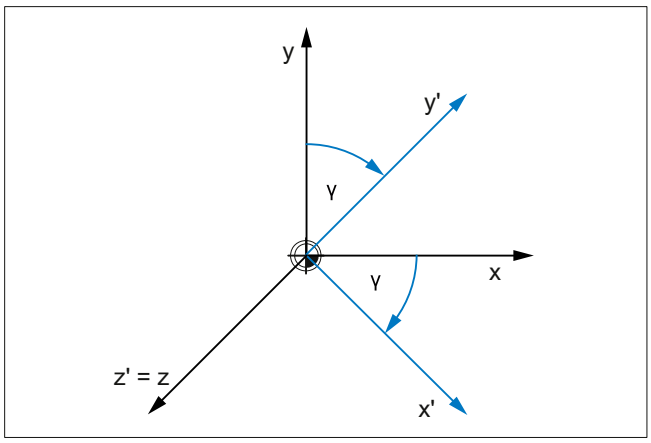
9.5.2.3 Rotation Overview (geometry axes only)

Function

The direction of rotation around the coordinate axes is determined by means of a right-hand, rectangular coordinate system with axes x, y and z. If the rotary motion is in a clockwise direction when looking in the positive direction of the coordinate axis, the direction of rotation is positive. A, B and C designate rotations whose axes are parallel to the coordinate axes.



The following figure shows the new position of the coordinate system x', y' and z' after the rotation around z with γ = -45°



Parameterization of the rotation sequence

The following machine data is used to set around which coordinate axes and in which order the rotations are performed when more than one angle of rotation is programmed:

MD10600 $MN_FRAME_ANGLE_INPUT_MODE = <value>

Value	Meaning
1	Euler angles in zy'x" convention (RPY angles)
2	Euler angles in zx'z" convention

Note

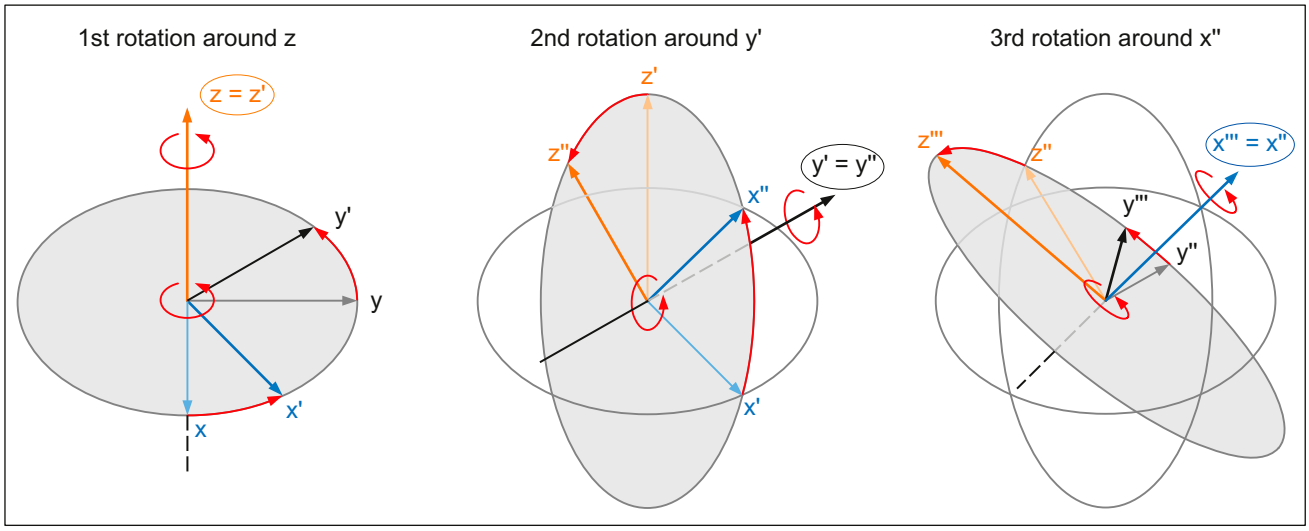
For historical reasons, Euler angles in zx'z" convention can be used. However, it is strongly recommended that only Euler angles in zy'x" convention (RPY angles) be used (see Section Rotation with a Euler angles: ZY'X" convention (RPY angles) (Page 768)).

9.5.2.4 Rotation with a Euler angles: ZY'X" convention (RPY angles)

Euler angles in zy'x" convention are also called RPY angles. RPY is derived from:

R: Roll → rotation around x
P: Pitch → rotation around y'
Y: Yaw → rotation around z"

With RPY angles, the rotations are in the order z, y', x".



Assignment of rotary axis to geometry axis

Rotary axis	Geometry axis in channel
x''	1st geometry axis
y'	2nd geometry axis
z	3rd geometry axis

Range of values

With RPY angles, programmed values can only be unambiguously calculated back within the following value ranges:

```
-180   <=   x   <=   180
 -90    <   y    <    90
-180   <=   z   <=   180
```

Programming of the complete frame

When programming the complete frame, all rotation components of the frame are always written. Non-programmed components are implicitly assigned the value 0°.

Syntax

```
<Frame> = CROT([<1st GAx>,<angle>,][<2nd GAx>,<angle>,][<3rd
GAx>,<angle>])
ROT [<1st GAx><angle>] [<2nd GAx><angle>] [<3rd GAx><angle>]
AROT [<1st GAx><angle>] [<2nd GAx><angle>] [<3rd GAx><angle>]
```

Meaning

CROT:	Absolute rotation, reference frame: Arbitrary programmed frame
<Frame>:	Arbitrary active or data management frame
ROT:	Absolute rotation, reference frame: Programmable frame $P_PFRAME, reference point: Zero point of the current workpiece coordinate system set with G54 ... G57, G505 ... G599
AROT:	Additive rotation, reference frame: Programmable frame $P_PFRAME, reference point: Zero point of the current workpiece coordinate system set with G54 ... G57, G505 ... G599
<nth GAx>:	Identifier of the nth geometry axis around which rotation is to be performed with the specified angle. The value 0° is implicitly set as angle of rotation for a geometry axis that has not been programmed. Assignment of geometry axis to rotary axis:

Geometry axis	Rotary axis
1st geometry axis	x''
2nd geometry axis	y'
3rd geometry axis	z

<angle>:	Angle specification in degrees.

Programming a frame component

When programming a frame component, only the programmed component of the frame is written. The components that are not programmed remain unchanged.

Syntax

```
<frame>[<index>,<GAx>,RT] = <angle>
```

Meaning

<Frame>:	Arbitrary active or data management frame
<Index>:	Array index of the frame, e.g. $P_UIFR[0 ... n]
<GAx>:	Identifier of the geometry axis around which rotation is to be performed with the specified angle.
RT:	Keyword for rotation "RoTation"
<Angle>	Angle specification in degrees.

Reading back the rotation components

In general, the same values are obtained when reading back the rotation components of a frame as those that were programmed:

Programmed	Saved rotation components 1)		
	x, RT	y, RT	z, RT
`<Frame>=CROT(X,45,Y,30,Z,-20)`	45	30	-20
1) The values of the saved rotation components are obtained when reading back			

Values outside the value range

Programmed values outside a value range are mapped on the range limits:

Programmed	Saved rotation components 1)		
	x, RT	y, RT	z, RT
`<Frame>=CROT(X,190,Y,0,Z,-200)`	-170	0	160
1) The values of the saved rotation components are obtained when reading back			

Note

It is recommended that when writing the rotation components of the frame, the specified value ranges are observed so that the same values are obtained when reading back the rotation components.

Gimbal lock

Gimbal lock designates a geometric problem in which the rotation components can no longer be unambiguously calculated back from the position vector. Gimbal lock occurs in RPY angles with an angular position of the rotation component y of 90°. In this case, the rotation components are converted by the controller so that the following applies:

- Rotation component x = 0°
- Rotation component y = 90°
- Rotation component z = resulting angle of rotation from: $z_{programmed}$ - $x_{programmed}$

Programmed	Saved rotation components		
	x, RT	y, RT	z, RT
`<Frame>=CROT(X,30,Y,90,Z,40)`	0	90	40 - 30 = 10

Differences when writing the complete frame and frame components

Two cases must be distinguished when writing the rotation components of a frame:

1. Writing the complete frame: `<Frame>=CROT(X,a,Y,b,Z,c)`

   When writing the complete frame, the conversion is immediately at the time of writing.

2. Writing individual rotation components, e.g. rotation around X: `<Frame>[0,X,RT]=a`

   When writing individual rotation components, the conversion depends on the storage location of the frame:

   – Data management frames

     With data management frames, the conversion is at the time of activation of the frame based on the rotation components written by this time. With regard to the conversion of a data management frame, a data management frame therefore behaves in the same way after writing individual rotation components as when writing the complete frame.

   – Active frames

     In the case of active frames, the conversion is immediately at time of writing of the rotation component.

Examples: Writing the complete frame

Programmed	Saved rotation components		
	x, RT	y, RT	z, RT
`<Frame>=CROT(X,0,Y,90,Z,90)`	0	90	90
`<Frame>=CROT(X,90,Y,90)`	0	90	**-90** 1)
`<Frame>=CROT(X,90,Y,90,Z,90)`	0	90	**0** 1)
1) Different values compared to the writing of **individual rotation components** of an **active frame**			

Examples: Writing individual rotation components of a data management frame

Programmed	Saved rotation components		
	x, RT	y, RT	z, RT

`<Data management frame>[0,X,RT]=0`
`<Data management frame>[0,Y,RT]=90`
`<Data management frame>[0,Z,RT]=90`
	0	90	90

`<Data management frame>[0,X,RT]=90`
`<Data management frame>[0,Y,RT]=90`
`<Data management frame>[0,Z,RT]=0`
	0	90	**-90** 1)

`<Data management frame>[0,X,RT]=90`
`<Data management frame>[0,Y,RT]=90`
`<Data management frame>[0,Z,RT]=90`
	0	90	**0** 1)

1) Different values compared to the writing of **individual rotation components** of an **active frame**			

Examples: Write individual rotation components of an active frame

Programmed	Saved rotation components		
	x, RT	y, RT	z, RT
<Active frame>[0,X,RT]=0 <Active frame>[0,Y,RT]=90 <Active frame>[0,Z,RT]=90	0	90	90
<Active frame>[0,X,RT]=90 <Active frame>[0,Y,RT]=90 <Active frame>[0,Z,RT]=0	0	90	**0** 1)
<Active frame>[0,X,RT]=90 <Active frame>[0,Y,RT]=90 <Active frame>[0,Z,RT]=90	0	90	**90** 1)
1) Different values compared to the writing of the **complete frame** or the writing of **individual rotation components** of a **data management frame**			

⚠ CAUTION

Because of the different conversion times after writing the **complete frame** or the writing of **individual rotation components** of a **data management frame** and the writing of **individual rotation components** of an **active frame**, different values can be read back for rotation component z.

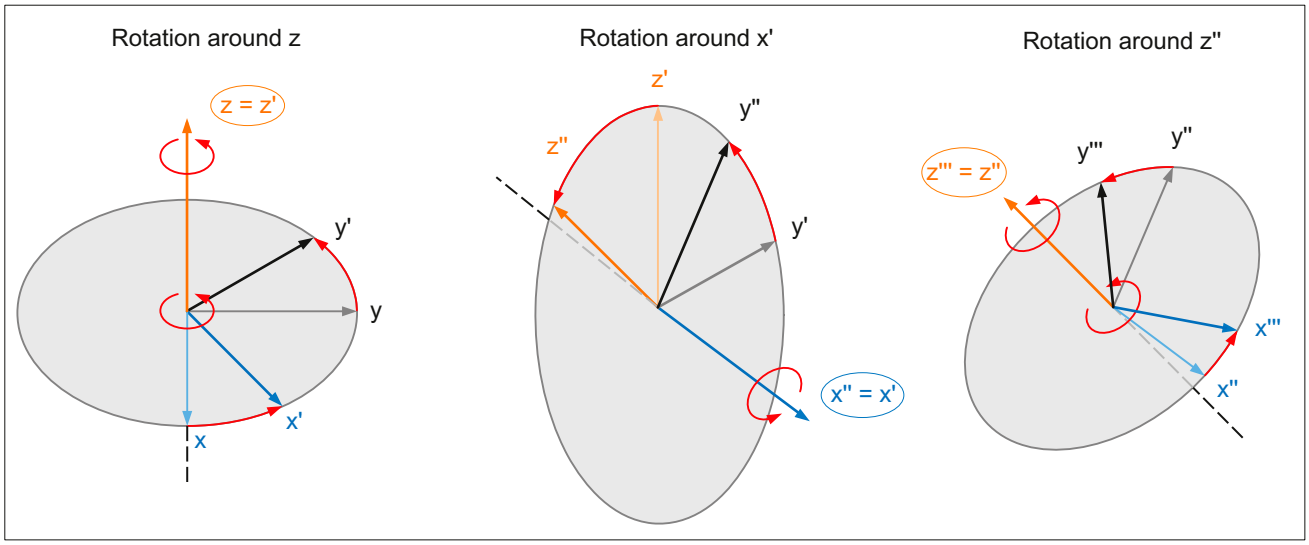
9.5.2.5 Rotation with a Euler angles: ZX'Z" convention

With Euler angles, the rotations are in the order z, x', z".

Note

Recommended use

For historical reasons, Euler angles in zx'z" convention can be used. However, it is strongly recommended that only Euler angles in zy'x" convention (RPY angles) be used (see Section Rotation with a Euler angles: ZY'X" convention (RPY angles) (Page 768)).



Assignment of rotary axis to geometry axis

Rotary axis	Geometry axis in channel
z''	3rd geometry axis
x'	1st geometry axis
z	3rd geometry axis

Range of values

Data from Euler angles can only be unambiguously calculated back within the following value ranges:

0	<=	**x**	<	180
-180	<=	**y**	<=	180
-180	<=	**z**	<=	180

For data outside the specified value ranges, a modulo conversion is made referred to the value of the particular range limit.

Note

It is recommended that when writing the rotation components of the frame, the specified value ranges are observed so that the same values are obtained when reading back the rotation components.

9.5.2.6 Rotation in any plane

CRPL - Constant Rotation Plane

The predefined function "**C**onstant **R**otation **Pl**ane" enables a rotation to be programmed for a frame in an arbitrary plane (G17, G18, G19) without specifying the identifier of a geometry axis. This enables rotations to be programmed in the third plane when only two geometry axes are present in the channel due to the specific machine constellation.

Syntax

```
CRPL(<rotary axis>,<angle of rotation>)
```

Meaning

CRPL: Rotation in any plane

<rotary axis>: Axis around which the rotation is performed

Type: INT

Value	Meaning
0	Rotation in the active plane
1	Rotation around Z
2	Rotation around Y
3	Rotation around X

<angle of rotation>: Angle in degrees through which the rotation is performed

Type: REAL

It is strongly recommended to observe the specified angular ranges. If the limits are not observed, then an unambiguous reverse calculation is not possible. Angles outside the limits are not rejected.

RPY angle:	X	-180 <= <angle of rotation> <= 180
	Y	-90 <= <angle of rotation> <= 90
	Z	-180 <= <angle of rotation> <= 180
ZX'Z" convention:	X	-180 <= <angle of rotation> <= 180
	Y	0 <= <angle of rotation> <= 180
	Z	-180 <= <angle of rotation> <= 180

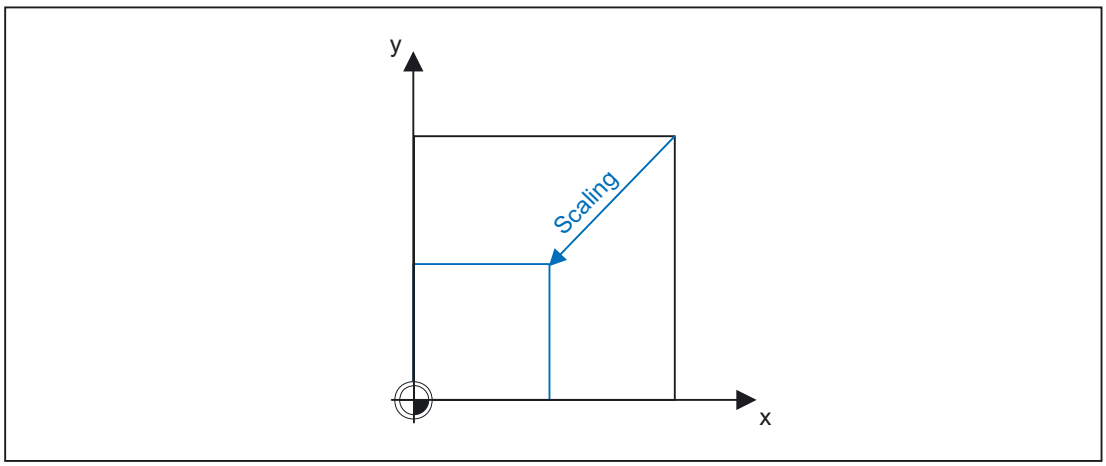
Chaining with frames

CRPL() can be chained with frames and known frame functions such as CTRANS(), CROT(), CMIRROR(), CSCALE(), CFINE() etc.

Examples:

```
$P_PFRAME = $P_PFRAME : CRPL(0,30.0)
$P_PFRAME = CTRANS(X,10) : CRPL(1,30.0)
$P_PFRAME = CROT(X,10) : CRPL(2,30.0)
$P_PFRAME = CRPL(3,30.0) : CMIRROR(Y)
```

9.5.2.7 Scaling

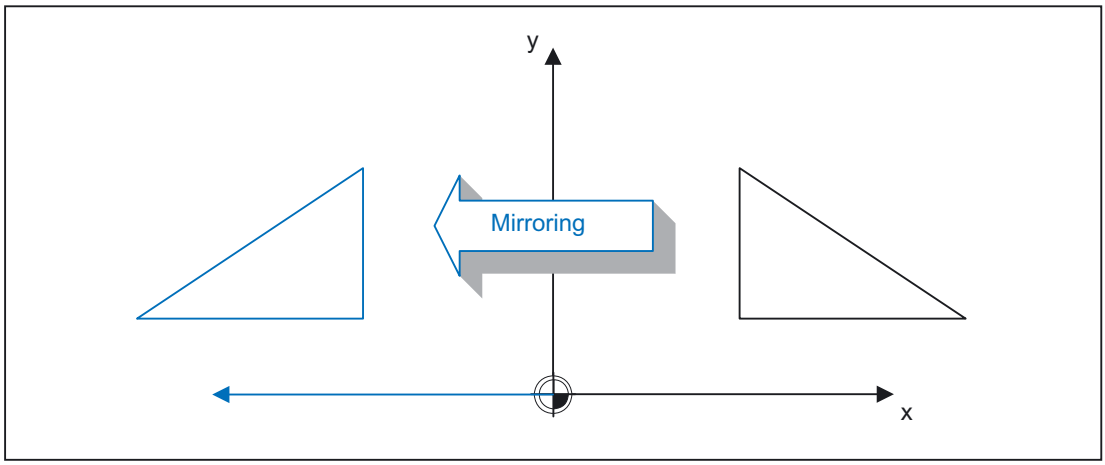


Programming

The program commands below are used to program the scaling:

```
$P_UIFR[1] = CSCALE(x,1,y,1)
SCALE x = 1y = 1
$P_UIFR[1,x,sc] = 1
```

9.5.2.8 Mirroring



Programming

The program commands below are used to program a mirroring:

```
$P_UIFR[1] = CMIRROR(x,1,y,1)

MIRROR x = 1y = 1

$P_UIFR[1,x,mi] = 1
```

9.5.2.9 Chain operator

Frame components or complete frames can be combined into a complete frame using the chain operator (:).

9.5.2.10 Programmable axis identifiers

Geo, channel and machine axis identifiers can be used in the frame commands. The programmed axis must be known to the channel-specific frames in the channel.

SPI

When programming frame instructions, the `SPI(<spindle number>)` axis function can be used in place of an axis identifier.

`SPI(<spindle number>)` forms the reference of the spindle to the channel axis.
→ refer to MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[ ] (assignment of spindle to machine axis)

The following frame instructions can be programmed with SPI(spino):

```
CTRANS()

CFINE()

CMIRROR()

CSCALE()
```

A spindle can only be assigned to one rotary axis at a time. The CROT(..) function can therefore not be programmed withSPI(), as only geometry axes are permitted forCROT().

The channel axis identifier or machine axis identifier of the axis belonging to the spindle is always output when decompiling frames, even when axis identifiers have been programmed in the part program with SPI(..).

If the spindle is assigned e.g., to the Channel Axis A then the programming:

```
N10 $P_UIFR[1] = CTRANS(SPI(1),33.33,X,1):CSCALE(SPI(1),33.33):CMIRROR(SPI(1))
```

during recompilation:

```
$P_UIFR[1]=CTRANS(X,1,A,33.33):CSCALE(A,33.33):CMIRROR(A)
```

If a spindle and an assigned axis are programmed in a frame instruction, then Alarm 16420 "Axis % multiply programmed" is output.

Example:

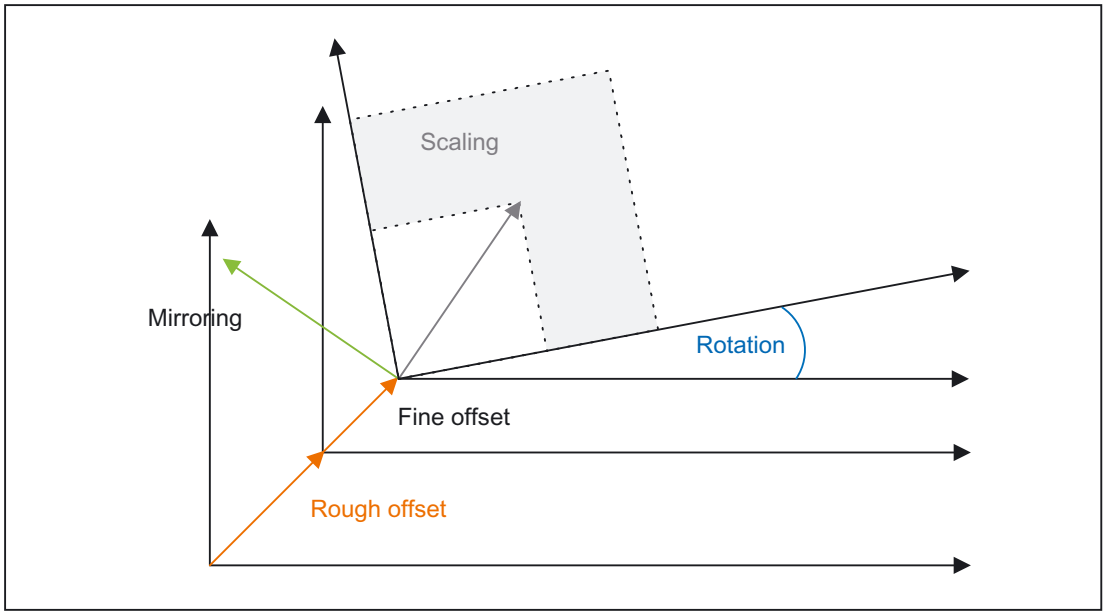
```
$P_UIFR[1] = CTRANS(SPI(1),33.33,X,1,A,44)
```

(The spindle is assigned to Axis A.)

Programming examples

```
$P_PFRAME[SPI(1),TR]=22.22

$P_PFRAME=CTRANS(X, axis value,Y,axis value,SPI(1),axis value)

$P_PFRAME=CSCALE(X,Scale,Y,scale,SPI(2),scale)

$P_PFRAME=CMIRROR(S1,Y,Z)

$P_UBFR=CTRANS(A,10):CFINE(SPI(1),0.1)
```

9.5.2.11 Coordinate transformation



The formulae below are used to discover the coordinate transformation for geometry axes:

WCS → BCS	$\vec{v} = R * \underline{S} * \underline{M} * \vec{v'} + \vec{t}$
BCS → WCS	$\vec{v'} = inv(\underline{M}) * inv(\underline{S}) * inv(R) + (\vec{v} - \vec{t})$

V: Position vector in BCS
V': Position vector in WCS

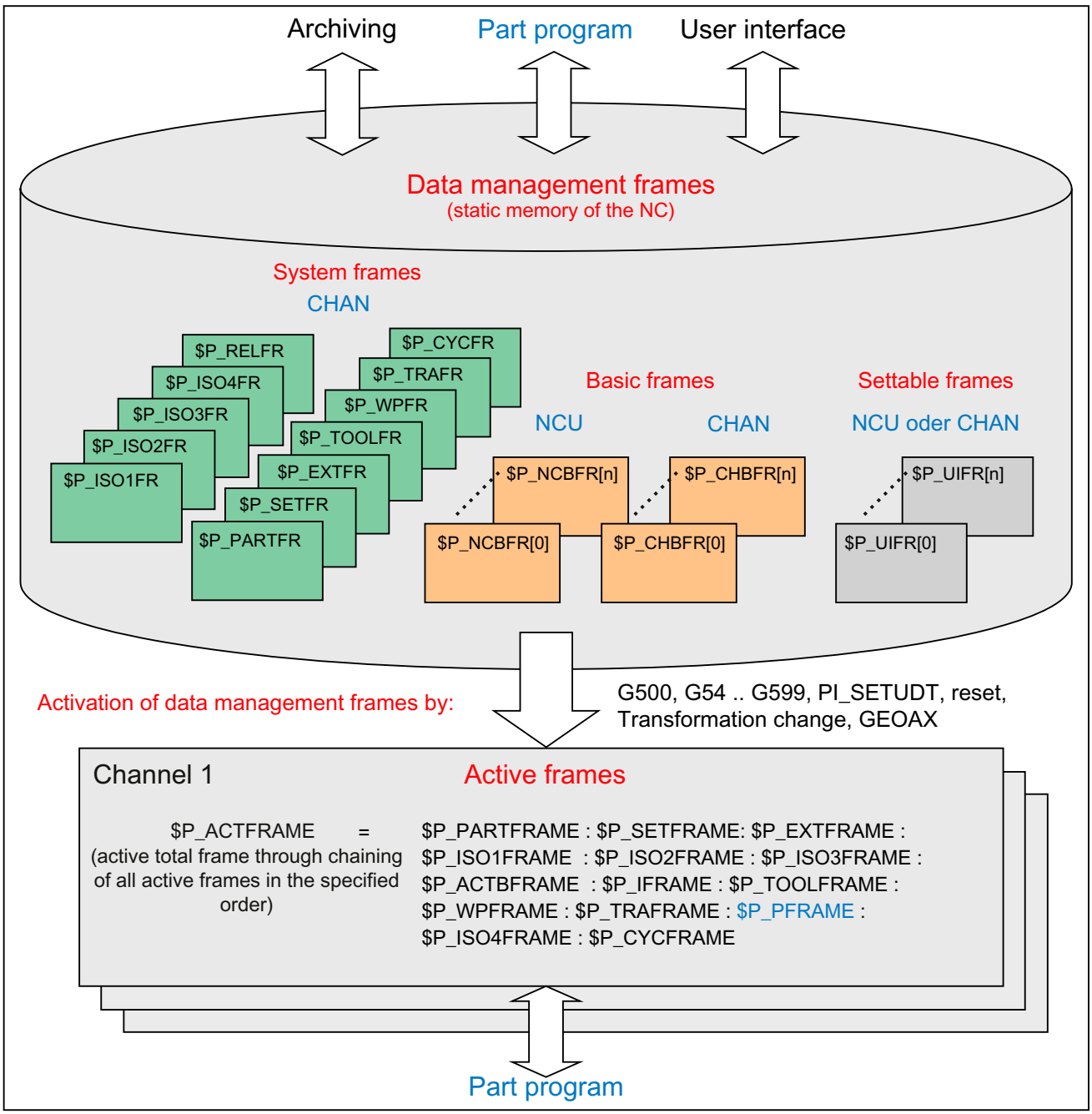
9.5.3 Frames in data management and active frames

9.5.3.1 Overview

The following frame types are available:

- System frames (see diagram)
- Basic frames (`$P_NCBFR[n]`, `$P_CHBFR[n]`)
- Adjustable frames (`$P_UIFR[n]`)
- Programmable frame (`$P_PFRAME[n]`)

Apart from the programmable frame, all types have a frame in the data management (data management frame) and an active frame. For a programmable frame, there is only one active frame.



Writing frames

Data management frames and active frames can be written from the part program. Only data management frames can be written via the user interface.

Archiving frames

Only data management frames can be archived.

9.5.3.2 Activating data management frames

Data management frames become active frames as a result of the following actions:

- Part program commands to activate/deactivate offsets: G54...G599, G500
- RESET and MD20110 $MC_RESET_MODE_MASK, Bit14 = 1
- Transformation changeover
- Changing the geometry axis assignment GEOAX
- Via SINUMERIK Operate with PI service "_N_SETUDT"

Activating via SINUMERIK Operate

The activation of a data management frame with PI service "_N_SETUDT" only becomes active in the channel after a hot restart for the selected part program. The activation is effective in the reset state if the following machine data is set:

MD9440 $MM_ACTIVATE_SEL_USER_DATA (set active offset immediately)

Activating system frames

System frames are activated by:

- programming the corresponding system function in the part program
- Operator control at SINUMERIK Operate

Note

Modifying system frames of the data management

Although in principle, system frames of the data management can be modified by the cycle programmer and activated using a G500, G54...G599 operation. However, this option should only be used with reservation.

Activating data management frames

The behavior when activating data management frames can be set using the following machine data:

MD24050 $MC_FRAME_SAA_MODE (save and activate data management frames)

Bit	Value	Meaning
0	0	Data management frames are activated **only** by programming the $P_CHBFRMASK, $P_NCBFRMASK and $P_CHSFRMASK bit masks. G500...G599 activate only the corresponding settable frame. The reset response is independent of this.
	1	Data management frames are not implicitly described by system functions, such as TOROT, PAROT, zero offset external and transformations.

System variable $P_CHSFRMASK

The system frames of the data management can be activated using system variable **$P_CHSFRMASK**. The value of the variables is specified as bit coded according to the machine data:
MD28082 $MC_MM_SYSTEM_FRAME_MASK (system frames of the data management)

The corresponding system frame of the data management in the channel is activated by setting a bit of the system variable **$P_CHSFRMASK** to a value of 1. For a value of 0, the currently active system frame in the channel remains active.

Activating system frames after RESET

After RESET, the system frames in the channel are activated whose bits are set in the following machine data:

MD24006 $MC_CHSFRAME_RESET_MASK (active system frames after Reset)

Activating system frames for TCARR, PAROT and TOROT, TOFRAME

The system frames for TCARR, PAROT and TOROT, TOFRAME are activated according to the setting in the following machine data:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

When changing over geometry axes using transformation selection/deselection or the GEOAX command, the actual total frame $P_ACTFRAME is either deleted or is re-calculated using the new geometry axis constellation and activated. The system frames and all other frames are conditioned again in relation to the geometry axes.

9.5.3.3 NCU global frames

All settable frames G54 to G599 and all basic frames can be configured NCU globally or channel-specifically. A combination of these is also possible with basic frames. Global frames affect all channels on an NCU. All channels have read and write access to the NCU. Global frames only have axial frame components, such as translations, scales and mirrors of individual axes. Each channel can read or modify global frames for any machine axis.

A characteristic of global frames is that they are calculated in all channels of an NCU. As the assignment of machine axes to channel axes and, in particular, to geometry axes, can be different in all channels, there is no geometric relationship. Global frames describe offsets, scales and mirrors of machine axes. Rotations cannot be used on global frames.

All settable frames can be reconfigured to global frames with the following machine data:

MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES (number of global, pre-defined user frames (SRAM))

If the value of this machine data is greater than zero, there are no channel-specific settable frames.

The following machine data becomes irrelevant then, and is not evaluated:

MD28080 $MC_MM_NUM_USER_FRAMES (number of settable frames (SRAM))

The number of global basic frames is parameterized through the following machine data:

MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES (number of global, basic frames (SRAM))

Channel-specific basic frames can also exist simultaneously through the following machine data:

MD28081 $MC_MM_NUM_BASE_FRAMES (number of basic frames (SRAM))

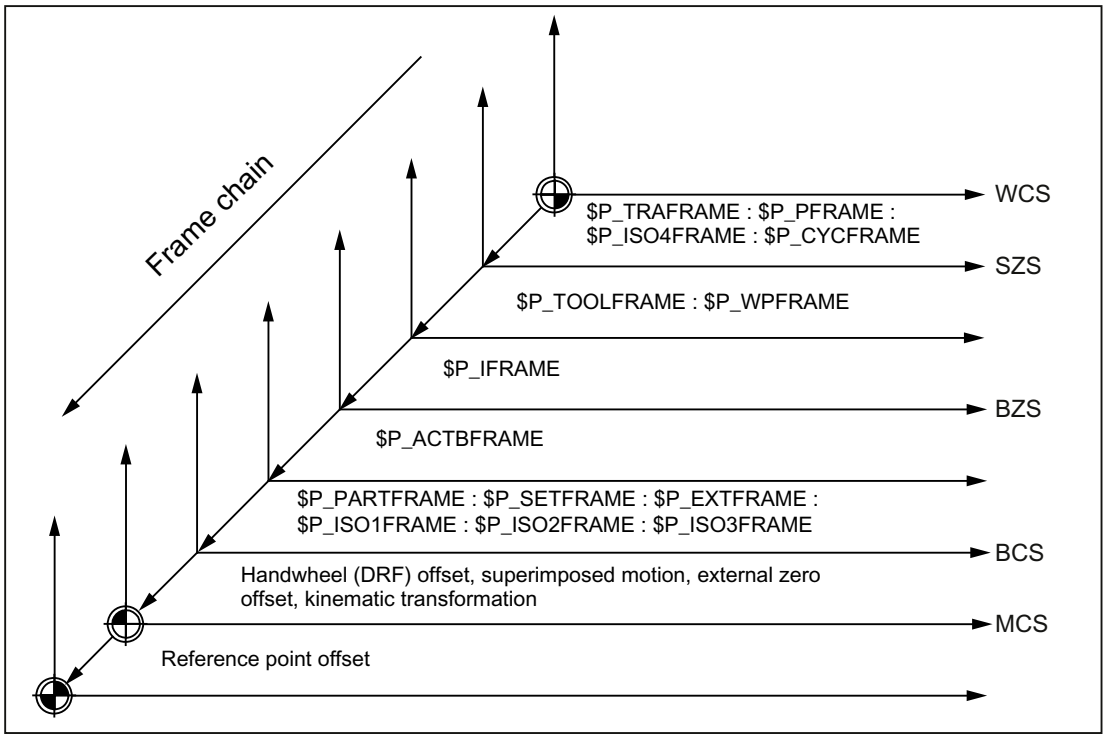
Global frames can be read and written from all channels of an NCU. When writing global frames, the user must ensure channel coordination. This can be achieved through e.g., wait markers.

9.5.4 Frame chain and coordinate systems

9.5.4.1 Overview

The figure below shows the frame chain for the current complete frame. The frame chain is located between the basic coordinate system (BCS) and the workpiece coordinate system (WCS).

The settable zero system (SZS) corresponds to the WCS transformed by the programmable frame. The basic zero system (BZS) still includes the current settable frame. The system frame for the external zero offset is only available if it has been configured, otherwise the external zero offset is traversed as an overlaid motion of the axis.



WCS: **W**orkpiece **C**oordinate **S**ystem
SZS: **S**ettable **Z**ero **S**ystem
BZS: **B**asic **Z**ero **S**ystem
BCS: **B**asic **C**oordinate **S**ystem
MCS: **M**achine **C**oordinate **S**ystem

The current complete frame `$P_ACTFRAME` results from the chaining of all active frames of the frame chain:

$P_ACTFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
$P_ACTBFRAME : $P_IFRAME : $P_TOOLFRAME :
$P_WPFRAME : $P_TRAFRAME: $P_PFRAME :
$P_ISO4FRAME : $P_CYCFRAME

9.5.4.2 Relative coordinate systems

Relative coordinate systems display the current setpoint positions of the axes which lie relative to a specified reference point in the active displayed coordinate system. No programming can be done regarding the relative coordinate systems. Only the axis positions in these systems can be read via the system variables.

The new display coordinate systems lie relative to WCS and ENS coordinate system and result through transformation of the WCS or ENS axis positions with the active system frame $P_RELFRAME. The relative coordinate systems can not only be displaced linearly, but also rotated, mirrored, compressed or expanded.

The position indicator for axis setpoints is done in WCS or in ENS. The configuring is done via HMI machine data. Always only one display-coordinate system is active in the channel. For this reason only one relative frame is provided which generates both relative coordinate systems in the same ratio. The HMI displays the relative coordinates according to the configuration.

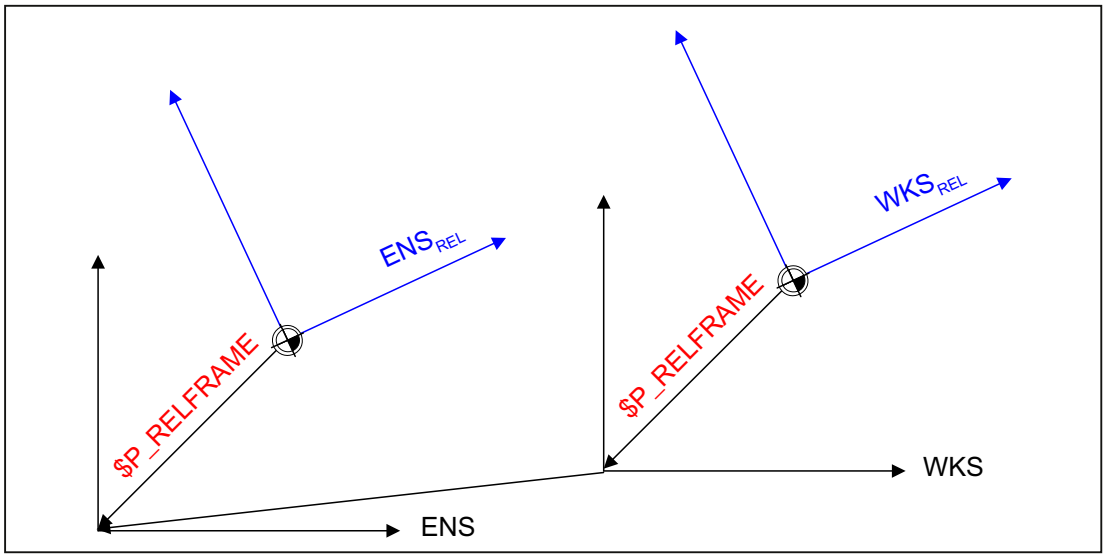


Figure 9-22 Relative coordinate systems

The data maintenance frame $P_RELFR can be written in the part program and via BTSS. All the frame components can be modified.

The active system frame $P_RELFRAME can be written in the part program and via BTSS.

The configuring of the system frame $P_RELFR is done via the following machine data:

Machine data	Bit	Meaning
MD28082 $MC_MM_SYSTEM_FRAME_MASK	11	Creation of $P_RELFR; with this, relative coordinate systems become existent.
MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK	11	Data maintenance frame $P_RELFR
MD24006 $MC_CHSFRAME_RESET_MASK.	11	$P_RELFR becomes active at Reset
MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK	11	$P_RELFR is deleted at Reset
MD24008 $MC_CHSFRAME_POWERON_MASK	11	$P_RELFR is deleted at PowerOn

The axis position in the relative coordinate system WCS_{Rel} can be read via the variable $AA_PCS_REL[ax]. The variable can be read in part program, BTSS and via synchronized actions.

The axis position in the relative coordinate system ENS_{Rel} can be read via the variable $AA_ACS_REL[ax]. The variable can be read in part program, BTSS and via synchronized actions.

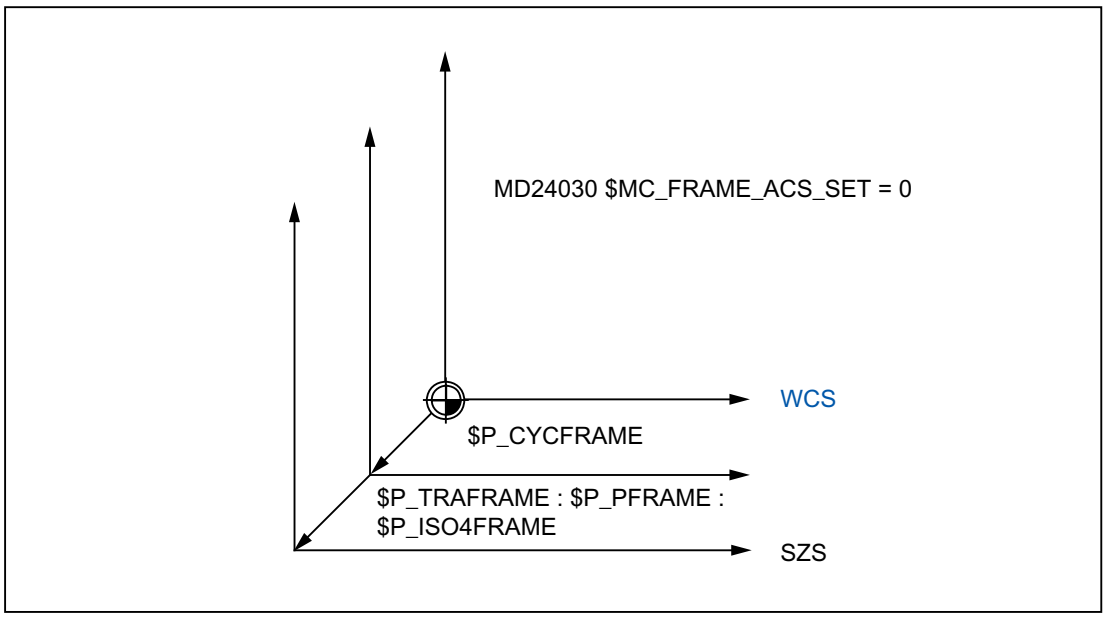
The setting of a relative reference point via the operator panel is done via the general command interface for the workpiece and tool measuring. The system frame $P_RELFR for relative coordinate systems is calculated and activated as follows:

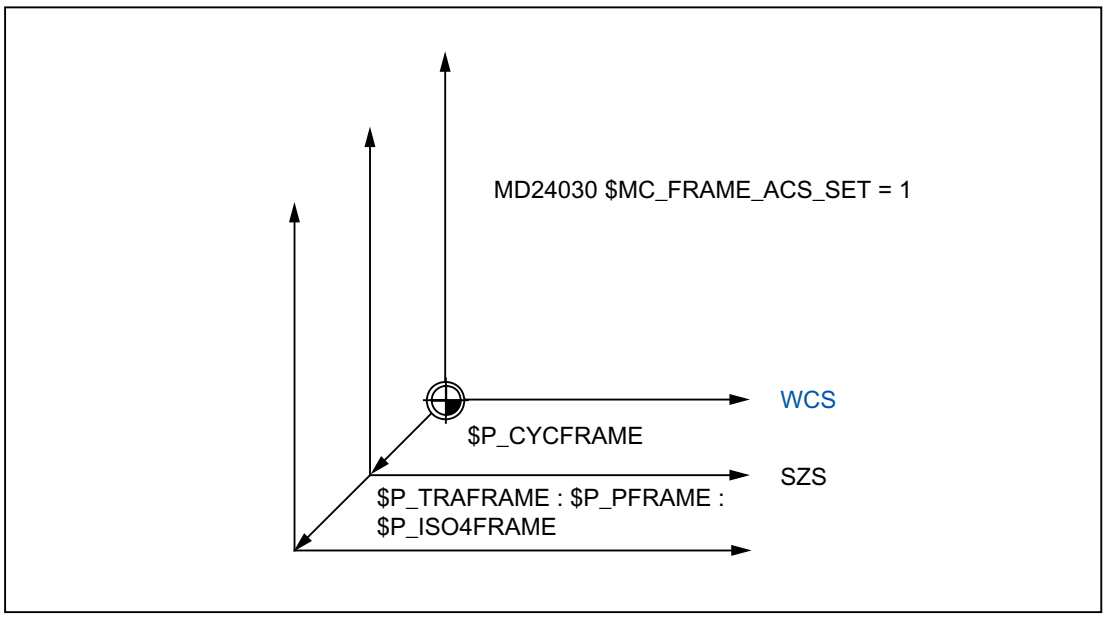
- $AC_MEAS_TYPE = 14
- PI-services _N_SETUDT(6, 7)

An example of setting the relative axis positions can be found in:
References:
Function Manual, Extended functions; Measurement (M5),
Section "Measurement of geometry and special axes (meas. type 14, 15)"

9.5.4.3 Configurable SZS

The function of the SZS coordinate system is to display actual values and move the axes during a cycle interruption. Cycles utilize frames in the frame chain to perform their functions. They input translations or rotations into either the programmable frame or the cycle system frame. The WCS is, therefore, modified by cycles. A user who uses Stop to interrupt a cycle, however, does not wish to traverse in the "cycle coordinate system", but in the programmed WCS. This is why the SZS is used for the display. For reasons of compatibility, the SZS is made configurable.





The following machine data can be used to set whether the ENS is with or without the programmable frame, the transformation frame and $P_ISO4FRAME:

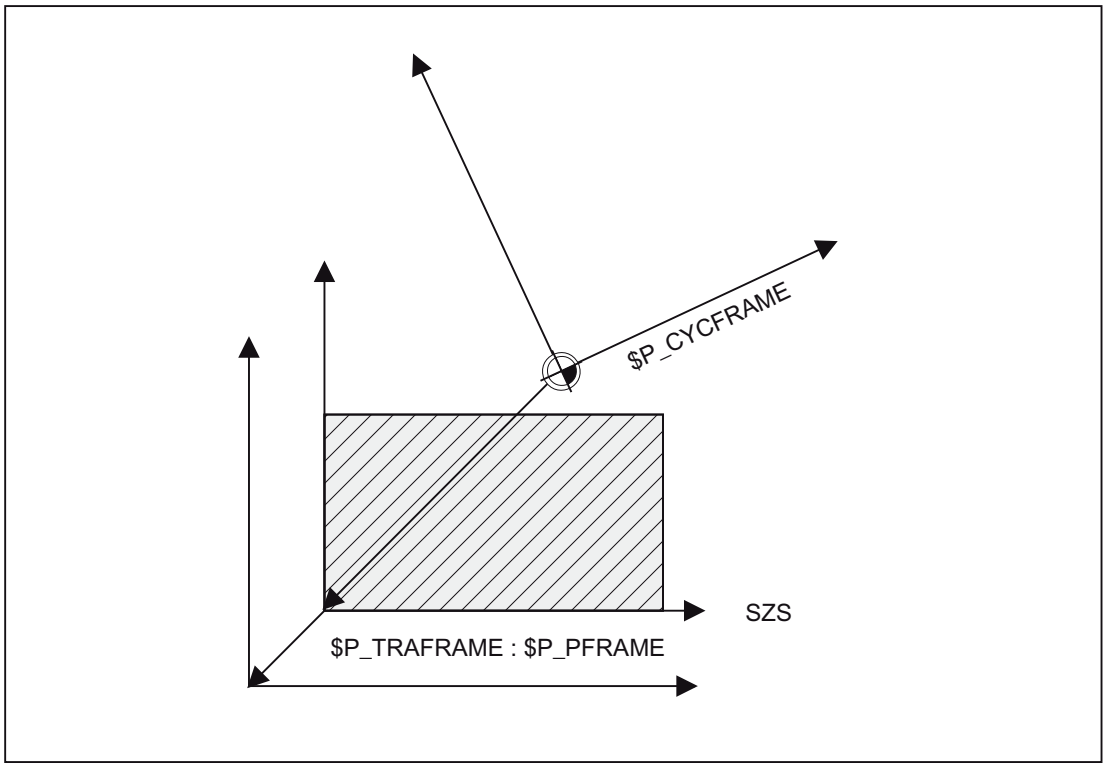
MD24030 $MC_FRAME_ACS_SET (setting of the ENS coordinate system)

As default, the value 1 is set.

Reconfiguring the SZS affects all SZS actual-value displays and the $AA_IEN[axis] system variables. Traversing geometry axes in JOG mode in the SZS also depends on the configuration.

9.5.4.4 Manual traverse in the SZS coordinate system

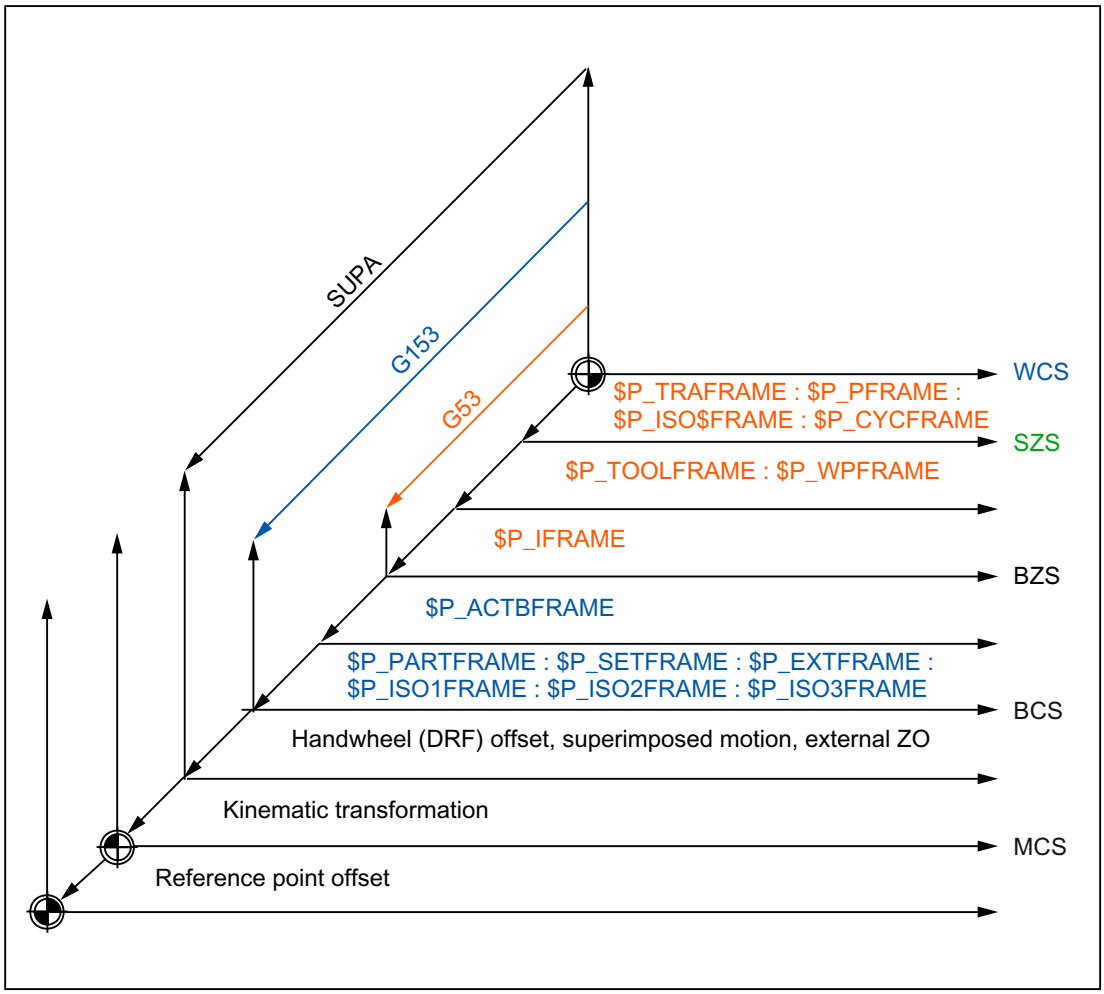
Previously, geometry axes have been traversed manually in JOG mode in the WCS. In addition, there is also the option to carry out this manual operation in the SZS coordinate system. The $AC_JOG_COORD variable enables the user to switch between manual traversing in the WCS and SZS. The user can now select if he wants to traverse in the SZS or the WCS.



9.5.4.5 Suppression of frames

Programming

Command	Meaning
G53	Nonmodal suppression of the following frames: • System frame for cycles • Programmable frame • System frame for transformations, workpieces, TOROT and TOFRAME • Active settable frame
G153	Nonmodal suppression of the following frames: • System frame for cycles • Programmable frame • System frame for TOROT and TOFRAME, workpieces • Active settable frame • All channel-specific and NCU global basic frames • System frames for PAROT, PRESET, scratching, ext. ZO
SUPA	Implicit preprocessing stop and non-modal suppression of frames analog G153 and additional • Handwheel offsets (DRF) • [Ext. zero offset] • Overlaid motion
G500	Modal activation of the G500 frame. The G500 frame should be a zero frame.
DRFOF	Deactivate (clear) the handwheel offsets (DRF)



Parameterization

Frame suppressions SUPA, G153 and G53 lead to the WCS, SZS and possibly the BZS jumping when frame suppression is active. This characteristic for position display and predefined position variables can be changed through the following machine data:

MD24020 $MC_FRAME_SUPPRESS_MODE (Positions during frame suppression)

Bit	Meaning
0	Positions for display (OPI) are without frame suppression.
1	Position variables are without frame suppression.

When the bit is set, the position for the display or the variables is calculated without frame suppression so that no further jumps in the position occur.

9.5.5 Frames of the frame chain

9.5.5.1 Overview

There are up to four frame variants:

- Settable frames (G500,G54 to G599)
- Basic frames
- Programmable frame
- System frames

9.5.5.2 Settable frames $P_UIFR[n]

The number of NCU global settable frames is set through the following machine data:

MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES (number of global, pre-defined user frames (SRAM))

The number can be between 0 and 100. If the MD has a value greater than zero, there are only NCU global settable frames, otherwise the following machine data specifies the number of channel-specific settable frames:

MD28080 $MC_MM_NUM_USER_FRAMES (number of settable frames (SRAM))

System variable $P_UIFR[n] can be used to read and write the frame field elements. The frame is not activated simultaneously when writing a field element, but rather activation only takes place on execution of a G500,G54, to G599 instruction. For NCU global frames, the changed frame only becomes active in those channels of the NCU, which execute a G500,G54 to G599 instruction. The variable is used primarily for storing write operations from HMI or PLC. These frame variables are saved by the data backup.

Current settable frame $P_IFRAME

The predefined frame variable $P_IFRAME can be used to read and write the current settable frame, which is valid in the channel, in the part program. The written settable frame is immediately included in the calculation. In the case of NCU global settable frames, the modified frame acts only in the channel in which the frame was programmed. If the frame is to be modified for all channels of an NCU, $P_UIFR[n] and $P_IFRAME must be written simultaneously. The other channels must then activate the corresponding frame, e.g., with G54.

Programming of settable frames

Settable frames can be read and written via the part program and via the OPI by operator actions and by the PLC. However, only data management frames can be written by the OPI. The index of the active settable frame can be ascertained via the $P_UIFRNUM system variable.

9.5.5.3 Channel basic frames $P_CHBFR[n]

The number of basic frames in the channel can be configured via the machine data:

MD28081 $MC_MM_NUM_BASE_FRAMES (number of basic frames (SRAM))

The minimum configuration is designed for at least one basic frame per channel. A maximum of 16 basic frames per channel is possible. In addition to the 16 basic frames, there can also be 16 NCU-global basic frames in the channel.

System variable $P_CHBFR[n] can be used to read and write the basic frame field elements. While writing a basic frame field element, the chained total frame is not activated. Instead, the activation takes place only after a `G500,G54..G599`instruction is executed. The variable is used primarily for storing write operations to the basic frame on HMI or PLC. These frame variables are saved by the data backup.

Current channel basic frames $P_CHBFRAME[n]

System variable $P_CHBFRAME[n] can be used to read and write the current channel basic frame field elements. The resulting total basic frame is calculated by means of the write process in the channel. Whenever a basic frame is written, the complete basic frame is calculated again.

Basic frame in channel $P_UBFR

The system variable is retained for reasons of compatibility, although it is redundant for the $P_CHBFR[0] variables.

The basic frame with field device 0 is not activated simultaneously when writing to the predefined $P_UBFR variable, but rather activation only takes place on execution of a `G500,G54,.G599`instruction. For NCU global frames, the changed frame only becomes active in those channels of the NCU, which execute a `G500,G54..G599`instruction. The variable is used primarily for storing write operations to the basic frame on HMI or PLC. The variable can also be read and written in the program.

$P_UBFR is identical to $P_CHBFR[0]. One basic frame always exists in the channel by default, so that the system variable is compatible with older versions. If there is no channel-specific basic frame, an alarm is issued at read/write: "Frame: Instruction not allowed" is output on a read or write access.

Current first basic frame in the channel $P_BFRAME

The system variable is retained for reasons of compatibility, although it is redundant for the $P_CHBFRAME[0] variables.

The predefined frame variable $P_BFRAME can be used to read and write the current basic frame with the field device of 0, which is valid in the channel, in the part program. The written basic frame is immediately included in the calculation. In the case of NCU global settable frames, the modified frame acts only in the channel in which the frame was programmed. If the frame is to be modified for all channels of an NCU, $P_UBFR and $P_BFRAME must be written simultaneously. The other channels must then activate the corresponding frame, e.g., with G54.

$P_BFRAME is identical to $P_CHBFRAME[0]. The system variable always has a valid default value. If there is no channel-specific basic frame, an alarm is issued at read/write: "Frame: Instruction not allowed" is output on a read or write access.

Programming basic frames

Basic frames can be read and written via the part program and via the OPI by operator actions and by the PLC. However, only data management frames can be written by the OPI.

9.5.5.4 NCU global basic frames $P_NCBFR[n]

The number of global basic frames can be configured via the machine data:

MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES (number of global, basic frames (SRAM))

There are a maximum of 16 global basic frames. All basic frames are stored as fields.

System variable $P_NCBFR[n] can be used to read and write the basic frame field elements. While writing a basic frame field element, the chained total frame is not activated. Instead, the activation takes place only after a G500,G54..G599instruction is executed. If the modified frame is to be active in every channel of the NCU, every channel must execute a G500,G54..G599instruction. The variable is used primarily for storing write operations to the basic frame on HMI or PLC. These frame variables are saved by the data backup.

Current NCU global basic frames $P_NCBFRAME[n]

System variable $P_NCBFRAME[n] can be used to read and write the current global basic frame field elements. The resulting total basic frame is calculated by means of the write process in the channel. The modified frame is activated only in the channel in which the frame was programmed. If the frame is to be modified for all channels of an NCU, $P_NCBFR[n] and $P_NCBFRAME[n] must be written simultaneously. The other channels must then activate the frame, e.g., with G54. Whenever a basic frame is written, the complete basic frame is calculated again.

Programming global frames

Global frames are programmed analogously, as are channel-specific frames, i.e., global basic frames are programmed with $P_NCBFR[n] and global settable frames with $P_UIFR[n].

Geometry axis, channel axis and machine axis identifiers can be used as axis identifiers for frame program commands. If there is no machine axis for the channel axis on the NCU, programming with channel axis identifiers is rejected with the alarm 18314 "Frame: Type conflict". Channel-specific frames can be programmed with geometry axis, channel axis and machine axis identifiers. If there is no corresponding channel axis for the machine axis on the NCU, programming with machine axis identifiers is rejected with the alarm 18314 "Frame: Type conflict". If frame components are applied to a machine axis or a channel axis, which is also a geometry axis, the corresponding geometry axis components will also be simultaneously modified.

Example:

```
$P_NCBFR[0] = CTRANS( ax1, 10 )
$P_NCBFR[0] = CTRANS(x, 10)
$P_NCBFR[0, ax1, FI ] = 0.1
$P_NCBFR[0, x, FI] = 0.1
```

Rotations cannot be used on global frames. The programming of a rotation is denied with alarm: "18310 Channel %1 Block %2 Frame: rotation not allowed" is displayed.

It is not possible to program chaining of global frames and channel-specific frames, and any attempt at this is rejected with the alarm 18314 "Frame: Type conflict". All global frames and channel-specific frames are internally chained to the complete frame. This takes place in the channel and only with all channel axes known in the channel. The assignment of a frame with rotation components to a global frame is denied with alarm "Frame: Rotation not allowed".

Example:

```
$P_NCBFR[0] = CTRANS( x, 10 ):CROT( y, 45    ; Faulty assignment on the global basic
)                                              frame
```

The following frames are channel-specific:

$P_UBFR, $P_BFRAME, $P_CHBFR[n],

$P_CHBFRAME[n], $P_NCBFRAME[n],

$P_ACTBFRAME and $P_ACTFRAME

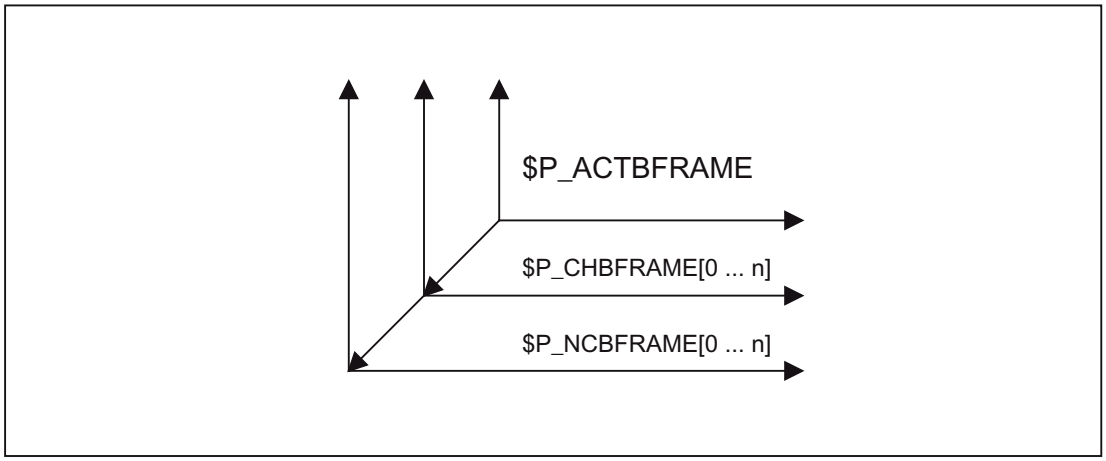
These frames can contain rotation components.

These frames only affect the channel that has been set.

With SW 5.1, attempts to program a channel axis, which is also a link axis, is rejected with alarm "14092 Channel %1 block %2 axis %3 is wrong axis type". An axis can be programmed only if it physically exists on the NCU.

9.5.5.5 Complete basic frame $P_ACTBFRAME

The chained complete basic frame is determined by the variable. The variable is read-only.



$P_ACTBFRAME=
$P_NCBFRAME[0] : ... : $P_NCBFRAME[n] : $P_CHBFRAME[0] : ... : $P_CHBFRAME[n]

Programmability of the complete basic frame

System variables $P_CHBFRMASK and $P_NCBFRMASK can be used to select which basic frames to include in the calculation of the "complete" basic frame. The variables can only be programmed in the program and read via the OPI. The value of the variables is interpreted as a bit mask and specifies which basic frame array element of $P_ACTBFRAME is included in the calculation. $P_CHBFRMASK can be used to define which channelspecific basic frames are included, and $P_NCBFRMASK can be used to define which NCU global basic frames are included in the calculation. When the variables are programmed, the complete basic frame and the complete frame are calculated again. After RESET and in the default setting, the value of $P_CHBFRMASK equals $MC_CHBFRAME_RESET_MASK and the value of $P_NCBFRMASK equals $MN_NCBFRAME_RESET_MASK.

```
$P_NCBFRMASK = 'H81'        ;   $P_NCBFRAME[0] : $P_NCBFRAME[7]
$P_CHBFRMASK = 'H11'        ;   $P_CHBFRAME[0] : $P_CHBFRAME[4]
```

9.5.5.6 Programmable frame $P_PFRAME

Programmable frames are available only as active frames.
This frame is reserved for the programmer.

The programmable frame remains at RESET, if:

MD24010 $MC_PFRAME_RESET_MODE (reset mode for programmable frame) = 1

This functionality is especially important after a RESET if one still wants to retract out of an oblique hole.

MIRROR

Mirrorings of a geometry axis were thus far (up to SW-P4) related to a defined reference axis only using the machine data:
MD10610 $MN_MIRROR_REF_AX
(reference axis for the mirroring).

From the user's point of view, this definition is hard to follow. When mirroring the z axis, the display showed that the x axis was mirrored and the y axis had been rotated about 180 degrees. When mirroring two axes this became even more complex and it was no longer easy to understand, which axes had been mirrored and, which had not.

With SW P5 and higher, there is the option to clearly display the mirroring of an axis. Mirroring is then not mapped to mirroring of a reference axis and rotations of other axes.

This setting can be configured using:

MD10610 $MN_MIRROR_REF_AX = 0

MIRROR and AMIRROR are used to expand the programming of the programmable frame. Previously, the specified value of the coordinate axis, e.g. the value 0 for MIRROR X0 is not evaluated, but the AMIRROR has a toggle function, i.e. MIRROR X0 activates mirroring and an additional AMIRROR X0 deactivates it. MIRROR always has an absolute effect and AMIRROR an additive effect.

The
MD10612 $MN_MIRROR_TOGGLE = 0 ("Mirror Toggle")
machine data setting can be used to define that the programmed values are evaluated.
A value of 0, as in AMIRROR X0, deactivates the mirroring of the axis, and values not equal to 0 cause the axis to be mirrored if it is not already mirrored.

Reading or writing mirroring component-by-component is independent of the machine data:

MD10612 $MN_MIRROR_TOGGLE

A value = 0 means that the axis is not mirrored and a value = 1 means that the axis will always be mirrored, irrespective of whether it has already been mirrored or not.

```
$P_NCBFR[0,x,mi]=1          ; the x axis is always mirrored.
$P_NCBFR[0,x,mi]=0          ; x axis mirroring off.
```

Axial replacement G58, G59 (only 840D sl)

The translation component of the programmable frame is split into an absolute component and a component for the total of all additively programmed translations. The absolute component can be changed using TRANS, CTRANS or by writing the translation components, in which the additive component is set to zero. G58 changes only the absolute translation component for the specified axis; the total of additively programmed translations is retained.

```
G58 X... Y... Z... A... ...
```

G59 is used to axially overwrite the additively programmed translations for the specified axes that were programmed with ATRANS.

```
G59 X... Y... Z... A... ...
```

Example:

```
TRANS X10 Y10 Z10
ATRANS X5 Y5                ; Total translations X15 Y15 Z10
G58 X20                     ; Total translations X25 Y15 Z10
G59 X10 Y10                 ; Total translations X30 Y20 Z10
```

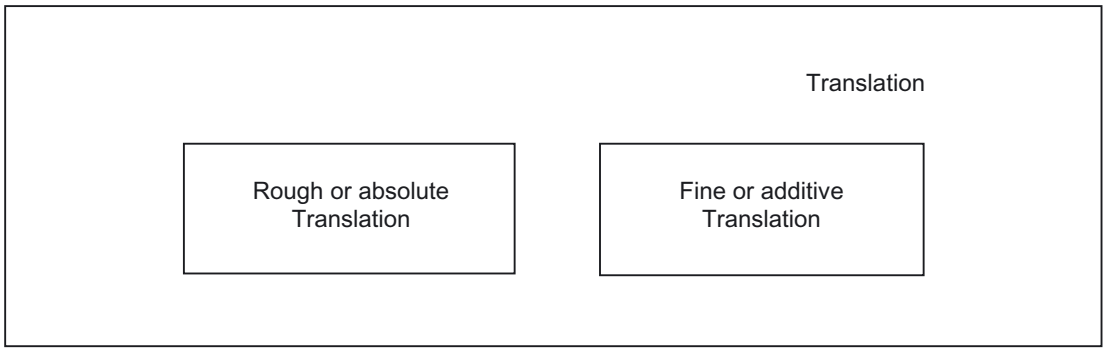
G58 and G59 can only be used if:

MD24000 $MC_FRAME_ADD_COMPONENTS (frame components for G58 / G59) = TRUE

Otherwise Alarm "18311 Channel %1 block %2 frame: instruction not permissible" is output.

The function can also only be used in conjunction with a configured fine offset for the programmable frame. If G58 or G59 is used without a configured fine offset, alarm "18312 channel %1 block %2 frame: Fine offset not configured" is output.

The absolute component of the translation is stored in the rough offset component and the additive translation component is stored in the fine offset component. To this end, the programmable frame or the fine offset is expanded.



The fine component is transferred on saving the programmable frame in a local frame variable (LUD or GUD) and on rewriting.

The table below shows the effect of various program commands on the absolute and additive translation.

	Coarse or absolute translation	Fine or additive translation
`TRANS X10`	10	0
`ATRANS X10`	Unchanged	alt_fine + 10
`CTRANS(X,10)`	10	0
`CTRANS()`	0	0
`CFINE(X,10)`	0	10
`$P_PFRAME[X,TR] = 10`	10	Unchanged
`$P_PFRAME[X,FI] = 10`	Unchanged	10
`G58 X10`	10	Unchanged
`G59 X10`	Unchanged	10

9.5.5.7 Channelspecific system frames

Channelspecific system frames

System frames are only described by system functions, such as PRESET, scratching, zero offset external and oblique processing. There are up to seven system frames per channel.

The valid system frames in the channel can be defined via machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK (System frames SRAM)

Only system frames required for system functions should be configured, in the interests of memory space. Per channel, each system frame occupies approx. 1 KB SRAM und approx. 6 KB DRAM. The system frame for PRESET and scratching and the system frame for cycles are the default. Channel-specific system frames are configured as bit codes, in accordance with the table below:

Bit	Default	System frame
0	1	PRESET and scratching
1	0	Zero offset external via system frames
2	0	TCARR and PAROT with an orientational toolholder
3	0	TOROT and TOFRAME
4	0	Frame for workpiece reference points
5	1	Frame for cycles
6	0	Frame for selection and deselection of transformations
7	0	$P_ISO1FRAME : Frame for G51.1 mirroring (ISO)
8	0	$P_ISO2FRAME : Frame for G68 2DROT (ISO)
9	0	$P_ISO3FRAME : Frame for G68 3DROT (ISO)
10	0	$P_ISO4FRAME: Frame for G51 scale (ISO)
11	0	$P_RELFR: Frame for relative coordinate systems

Example:
$MC_MM_SYSTEM_FRAME_MASK = 'B001101' means that
there are three system frames; one for PRESET, one for PAROT and one for TOROT and TOFRAME.

The system frame mask is used to define if the corresponding function has a system frame. With non-configured frames, in certain circumstances the function will be rejected with an alarm.

System frames in data management

The system frames are stored in the static NC memory and can, therefore, be archived and reloaded. System frames in data management can be read and written in the program using the following variables:

System variables	Significance
$P_SETFR	System frame for PRESET and scratching (**SetFr**ame)
$P_EXTFR	System frame for zero offset external (**ExtFr**ame)
$P_PARTFR	System frame for TCARR and PAROT (**PartFr**ame)
$P_TOOLFR	System frame for TOROT and TOFRAME (**ToolFr**ame)
$P_WPFR	System frame for workpiece reference points (**W**ork-**P**iece-**Fr**ame)
$P_CYCFR	System frame for cycles (**Cyc**le-**Fr**ame)
$P_TRAFRAME	System frame for transformations (**Tran**sformation **Fr**ame)
$P_ISO1FR	Frame for G51.1 mirroring (ISO)
$P_ISO2FR	Frame for G68 2DROT (ISO)
$P_ISO3FR	Frame for G68 3DROT (ISO)
$P_ISO4FR	System frame for G51 scale (ISO)
$P_RELFR	System frame for relative coordinate systems

All write operations to these frames must be executed using system functions. For cycle programmers, it has been made possible to write the frames using the above variables. Attempts to write to a non-configured system frame are rejected with the alarm "Channel %1 block %2 name %3 not defined or option not available".

System frames in the data management are either activated directly with the system function (TOROT, PAROT, etc.), or with a G500, G54 to G599 instruction.

Active system frames

The active system frames are the frames, which are active in the main run. An appropriate current system frame exists for each current system frame in the data management. Only with the activation of the data management frame are the values taken into account with regard to the preprocessing.

The following current system frames exist:

- $P_SETFRAME

  In the part program, the variable $P_SETFRAME can be used to read and write the current system frame for PRESET and scratching. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_EXTFRAME

  In the part program, the variable $P_EXTFRAME can be used to read and write the current system frame for the zero offset external. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_PARTFRAME

  In the part program, the variable $P_PARTFRAME can be used to read and write the current system frame for TCARR and PAROT for toolholders with orientation capability. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_TOOLFRAME

  In the part program, the variable $P_TOOLFRAME can be used to read and write the current system frame for TOROT and TOFRAME. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_WPFRAME

  In the part program, the variable $P_WPFRAME can be used to read and write the current system frame for setting workpiece reference points. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_CYCFRAME

  In the part program, the variable $P_CYCFRAME can be used to read and write the current system frame for cycles. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_TRAFRAME

  In the part program, the variable $P_TRAFRAME can be used to read and write the current system frame for transformations. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_ISO1FRAME, $P_ISO2FRAME, $P_ISO3FRAME, $P_ISO4FRAME

  One can read and write the current system frames for special ISO language commands in the parts program through the variables. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_RELFRAME

  In the part program, the variable $P_RELFRAME can be used to read and write the current system frame for relative coordinate systems. The variable returns a zero frame if the system frame is not configured through MD28082.

- $P_ACSFRAME

  The currently resulting frame that is defined by the ENS-(ACS) coordinate system, can be read and written through the $P_ACSFRAME variable.

For MD24030 $MC_FRAME_ACS_SET = 0, the frame is calculated as follows:

$P_ACSFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
$P_ACTBFRAME : $P_IFRAME :
$P_TOOLFRAME : $P_WPFRAME

For MD24030 $MC_FRAME_ACS_SET = 1, the frame is calculated as follows:

$P_ACSFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
$P_ACTBFRAME : $P_IFRAME :
$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME:
$P_PFRAME : $P_ISO4FRAME

- $P_ACTFRAME

  The resulting current complete frame $P_ACTFRAME is now a chain of all system frames, basic frames, the current settable frame and the programmable frame. The current frame is always updated whenever a frame component is changed.

  The current complete frame is calculated according to the formula below:

  $P_ACTFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
  $P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
  $P_ACTBFRAME : $P_IFRAME :
  $P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME:
  $P_PFRAME : $P_ISO4FRAME : $P_CYCFRAME

9.5.6 Implicit frame changes

9.5.6.1 Frames and switchover of geometry axes

In the channel, the geometry axis configuration can be changed by switching a transformation on and off and with the GEOAX() command (R3).

Machine data
MD10602 $MN_FRAME_GEOAX_CHANGE_MODE
can be used to configure, for all channels of the system, whether the current complete frame is calculated again on the basis of the new geometry axes or whether the complete frame is deleted.

Four modes can be set via machine data:

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 0

  The current complete frame is deleted when geometry axes are switched over, when transformations are selected and deselected, and on GEOAX().
  The modified geometry axis configuration is not used until a new frame is activated.

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

  The current complete frame is calculated again when the geometry axes are switched over, and the translations, scalings and mirrorings of the new geometry axes are effective. The rotations of the geometry axes which were programmed before the switchover remain effective for the new geometry axes.
  For TRANSMIT, TRACYL and TRAANG, see Section "Frame for selection and deselection of transformations (Page 807)".

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

  The current complete frame is calculated again when the geometry axes are switched over, and the translations, scalings and mirrorings of the new geometry axes are effective. If rotations are active in the current basic frames, the current settable frame or the programmable frame before the switchover, it is aborted with the alarm "Frame: Geometry axis switchover not allowed".
  For TRANSMIT, TRACYL and TRAANG, see Section "Frame for selection and deselection of transformations (Page 807)".

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 3

  The current frame is deleted when selecting and deselecting transformations.
  With GEOAX(), the current complete frame is calculated again and the translations, scalings and mirrorings of the new geometry axes come into effect.
  The rotations of the geometry axes which were programmed before the switchover remain effective for the new geometry axes.

The workpiece geometry is described by a coordinate system that is formed by the geometry axes. A channel axis is assigned to each geometry axis and a machine axis is assigned to each channel axis. An axial frame exists for each machine axis and for each frame (system frame, basic frame, settable frame, programmable frame). When a new machine axis is assigned to a geometry axis, the axial frame components of the machine axis, such as translations (coarse and fine), scaling and mirroring of the appropriate frame, are also applied. The new geometry in the channel is then generated by the new contour frames resulting from the new geometry axes (up to three in number).

The current valid frames are calculated again on the geometry axis switchover and a resulting complete frame is generated. The data management frames are not included unless they are activated.

Example:

The channel axis is to become a geometry axis x through `geo axis` substitution. The substitution is to give the programmable frame a translation component of 10 in the x axis. The current settable frame is to be retained.

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

Program code	Comment
`$P_UIFR[1] = CROT(x,10,y,20,z,30)`	`; Frame is retained after geo axis substitution.`
`G54`	`; Settable frame becomes active.`
`TRANS a10`	`; Axial offset of a is also substituted.`
`GEOAX(1, a)`	`; a becomes x axis;` `$P_ACTFRAME=CROT(x,10, y,20,z,30):CTRANS(x10).`

Several channel axes can become geometry axes on a transformation change.

Example:

Channel axes 4, 5 and 6 become the geometry axes of a 5axis transformation. The geometry axes are thus all substituted before the transformation. The current frames are changed when the transformation is activated. The axial frame components of the channel axes which become geometry axes are taken into account when calculating the new WCS. Rotations programmed before the transformation are retained. The old WCS is restored when the transformation is deactivated. The most common application will be that the geometry axes do not change before and after the transformation and that the frames are to stay as they were before the transformation.

Machine data:

```
$MN_FRAME_GEOAX_CHANGE_MODE = 1

$MC_AXCONF_CHANAX_NAME_TAB[0] = "CAX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "CAY"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "CAZ"
$MC_AXCONF_CHANAX_NAME_TAB[3] = "A"
$MC_AXCONF_CHANAX_NAME_TAB[4] = "B"
$MC_AXCONF_CHANAX_NAME_TAB[5] = "C"

$MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
$MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2
$MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"
$MC_AXCONF_GEOAX_NAME_TAB[1]="Y"
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=5
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=6

$MC_TRAFO_AXES_IN_1[0] = 4
$MC_TRAFO_AXES_IN_1[1] = 5
$MC_TRAFO_AXES_IN_1[2] = 6
$MC_TRAFO_AXES_IN_1[3] = 1
$MC_TRAFO_AXES_IN_1[4] = 2
```

Program:

```
Program code
$P_NCBFRAME[0] = ctrans(x,1,y,2,z,3,a,4,b,5,c,6)
$P_CHBFRAME[0] = ctrans(x,1,y,2,z,3,a,4,b,5,c,6)
$P_IFRAME = ctrans(x,1,y,2,z,3,a,4,b,5,c,6):crot(z,45)
$P_PFRAME = ctrans(x,1,y,2,z,3,a,4,b,5,c,6):crot(x,10,y,20,z,30)
```

```
Program code        Comment
TRAORI              ; Transformation sets GeoAx(4,5,6)
                    ; $P_NCBFRAME[0] = ctrans(x,4,y,5,z,6,cax,1,cay,2,caz,3)
                    ; $P_ACTBFRAME =ctrans(x,8,y,10,z,12,cax,2,cay,4,caz,6)
                    ; $P_PFRAME =
                      ctrans(x,4,y,5,z,6,cax,1,cay,2,caz,3):crot(x,10,y,20,z,30)
                    ; $P_IFRAME = ctrans(x,4,y,5,z,6,cax,1,cay,2,caz,3):crot(z,45)
TRAFOOF             ; Deactivation of the transformation sets GeoAx(1,2,3)
                    ; $P_NCBFRAME[0] = ctrans(x,1,y,2,z,3,a,4,b,5,c,6)
                    ; $P_CHBFRAME[0] = ctrans(x,1,y,2,z,3,a,4,b,5,c,6)
                    ; $P_IFRAME = ctrans(x,1,y,2,z,3,a,4,b,5,c,6):crot(z,45)
                    ; $P_PFRAME =
                      ctrans(x,1,y,2,z,3,a,4,b,5,c,6):crot(x,10,y,20,z,30)
```

9.5.6.2 Frame for selection and deselection of transformations

This function is available with NCK 51.00.00 and higher. Transformations TRANSMIT, TRACYL and TRAANG are supported.

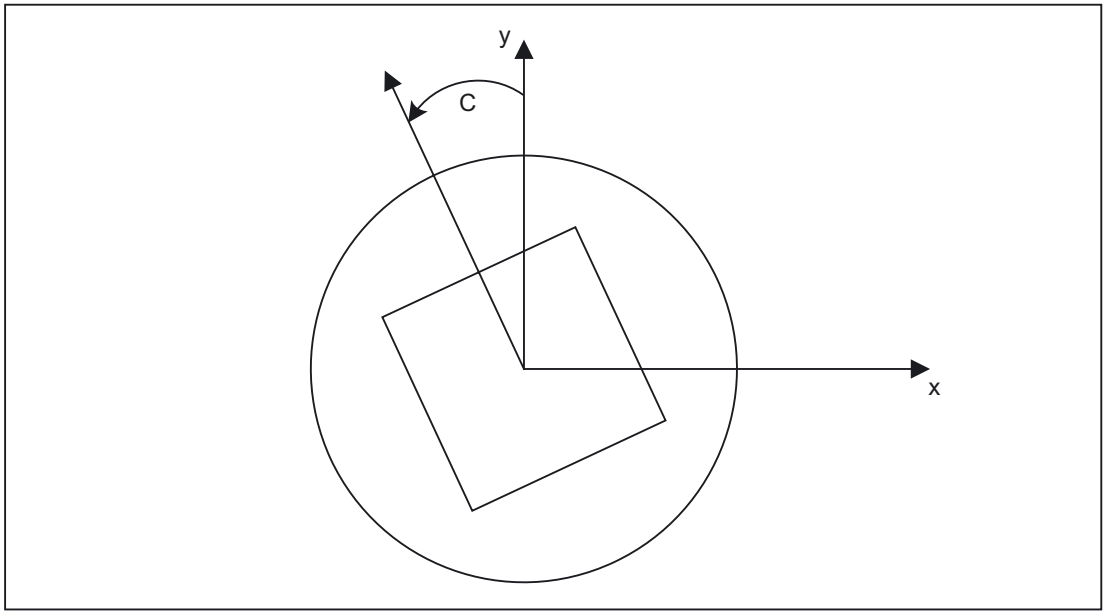
As a rule, the assignment of geometry axes to channel axes changes when selecting and deselecting transformations. It is not possible to uniquely assign axial frame components to geometric contour frame components when carrying out transformations, in which rotary axes become linear axes and vice versa. The contour frame must be conditioned using special treatment for such non-linear transformations.

The mode, set with MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1 and 2, is expanded in such a way as to take the above transformations into account.

When selecting transformations, the contour frame is connected to the axial frames. With transformations TRANSMIT, TRACYL and TRAANG, the virtual geometry axis is subject to special treatment.

NOTICE
Transformations with virtual axes When selecting TRANSMIT or TRACYL, offsets, scaling and mirroring of the real Y axis are not transferred and accepted in the virtual Y axis. Offsets, scaling and mirroring of the virtual Y axis are deleted for TRAFOOF.

TRANSMIT



Transmit expansions:

The machine data below can be used to take the axial complete frame of the TRANSMIT rotary axis, i.e., the translation, fine offset, mirroring and scaling, into account in the transformation:

MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 1

MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 1

A rotary axis offset can, for example, be entered by compensating the oblique position of a workpiece in a frame within a frame chain. As a rule, this offset can also be included in the transformation as an offset in the rotary axis. A c axis offset, as in the figure above, then leads to corresponding x and y values.

MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 2

MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 2

With this setting, the axial offset of the rotary axis is taken account of in the transformation up to the SZS. The axial offsets of the rotary axis included in the SZS frames are entered into the transformation frame as rotation. This setting is only effective if the transformation frame has been configured.

Frame expansions:

The expansions described below are only valid for the following machine data settings:

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

The selection of transformation TRANSMIT produces a virtual geometry axis, coupled by way of the rotary axis, which is merely included in the contour frame but does not have a reference to an axial frame. The geometric value results from the rotation of a rotary axis. All other geometry axes accept their axial components when the transformation is selected.

Translations:

When selecting TRANSMIT, translations of the virtual axis are deleted. Translations of the rotary axis can be taken into account in the transformation.

Rotations:

Rotation before the transformation is taken over.

Mirroring:

Mirroring of the virtual axis is deleted. Mirroring of the rotary axis can be taken into account in the transformation.

Scaling:

Scaling of the virtual axis is deleted. Scaling of the rotary axis can be taken into account in the transformation.

Example:

Machine data for TRANSMIT

```
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK='H41'              ; TRAFRAME, SETFRAME
$MC_CHSFRAME_RESET_MASK='H41'               ; Frames are active after Reset.
$MC_CHSFRAME_POWERON_MASK='H41'             ; Frames are deleted for Power On.

$MN_FRAME_GEOAX_CHANGE_MODE=1               ; Frames are calculated after switchover
                                              of the geo axis.
$MC_RESET_MODE_MASK='H4041'                 ; Basic frame is not deselected after
                                              RESET.
;$MC_RESET_MODE_MASK='H41'                  ; Basic frame is deselected after RESET.

;$MC_GCODE_RESET_VALUES[7]=2                ; G54 is the default setting.
$MC_GCODE_RESET_VALUES[7]=1                 ; G500 is the default setting.

$MN_MM_NUM_GLOBAL_USER_FRAMES=0
$MN_MM_NUM_GLOBAL_BASE_FRAMES=3

$MC_MM_NUM_USER_FRAMES=10                   ; from 5 to 100
$MC_MM_NUM_BASE_FRAMES=3                    ; from 0 to 8

$MN_NCBFRAME_RESET_MASK='HFF'
$MC_CHBFRAME_RESET_MASK='HFF'

$MN_MIRROR_REF_AX=0                         ; No scaling when mirroring.
$MN_MIRROR_TOGGLE=0
$MN_MM_FRAME_FINE_TRANS=1                   ; Fine offset
$MC_FRAME_ADD_COMPONENTS=TRUE               ; G58, G59 is possible.
```

```
; TRANSMIT is 1st transformer

$MC_TRAFO_TYPE_1=256

$MC_TRAFO_AXES_IN_1[0]=1
$MC_TRAFO_AXES_IN_1[1]=6
$MC_TRAFO_AXES_IN_1[2]=3
$MC_TRAFO_AXES_IN_1[3]=0
$MC_TRAFO_AXES_IN_1[4]=0

$MA_ROT_IS_MODULO[AX6]=TRUE;

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=6
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3

$MC_TRANSMIT_BASE_TOOL_1[0]=0.0
$MC_TRANSMIT_BASE_TOOL_1[1]=0.0
$MC_TRANSMIT_BASE_TOOL_1[2]=0.0

$MC_TRANSMIT_ROT_AX_OFFSET_1=0.0
$MC_TRANSMIT_ROT_SIGN_IS_PLUS_1=TRUE

$MC_TRANSMIT_ROT_AX_FRAME_1=1
```

```
; TRANSMIT is 2nd transformer

$MC_TRAFO_TYPE_2=256

$MC_TRAFO_AXES_IN_2[0]=1
$MC_TRAFO_AXES_IN_2[1]=6
$MC_TRAFO_AXES_IN_2[2]=2
$MC_TRAFO_AXES_IN_2[3]=0
$MC_TRAFO_AXES_IN_2[4]=0

$MC_TRAFO_GEOAX_ASSIGN_TAB_2[0]=1
$MC_TRAFO_GEOAX_ASSIGN_TAB_2[1]=6
$MC_TRAFO_GEOAX_ASSIGN_TAB_2[2]=2

$MC_TRANSMIT_BASE_TOOL_2[0]=4.0
$MC_TRANSMIT_BASE_TOOL_2[1]=0.0
$MC_TRANSMIT_BASE_TOOL_2[2]=0.0

$MC_TRANSMIT_ROT_AX_OFFSET_2=19.0
$MC_TRANSMIT_ROT_SIGN_IS_PLUS_2=TRUE

$MC_TRANSMIT_ROT_AX_FRAME_2=1
```

Part program:

```
; Frame settings
N820 $P_UIFR[1] = ctrans(x,1,y,2,z,3,c,4)
N830 $P_UIFR[1] = $P_UIFR[1] : crot(x,10,y,20,z,30)
N840 $P_UIFR[1] = $P_UIFR[1] : cmirror(x,c)
N850
N860 $P_CHBFR[0] = ctrans(x,10,y,20,z,30,c,15)
N870
```

```
; Tool selection, clamping compensation, plane selection
N890 T2 D1 G54 G17 G90 F5000 G64 SOFT
N900
```

```
; Approach start position
N920 G0 X20 Z10
N930
N940 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,C,15)
N950 setal(61000)
N960 endif
N970 if $P_BFRAME <> $P_CHBFR[0]
N980 setal(61000)
N990 endif
N1000 if $P_IFRAME <>
CTRANS(X,1,Y,2,Z,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1010 setal(61000)
N1020 endif
N1030 if $P_IFRAME <> $P_UIFR[1]
N1040 setal(61000)
N1050 endif
N1060 if $P_ACTFRAME <>
CTRANS(X,11,Y,22,Z,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1070 setal(61000)
N1080 endif
N1090
N1100 TRANSMIT(2)
N1110
N1120 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,20,CAZ,30,C,15)
N1130 setal(61000)
N1140 endif
N1180 if $P_IFRAME <>
CTRANS(X,1,Y,0,Z,2,CAZ,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1190 setal(61000)
N1200 endif
N1240 if $P_ACTFRAME <>
CTRANS(X,11,Y,0,Z,22,CAZ,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1250 setal(61001)
N1260 endif
N1270
N1280
N1290 $P_UIFR[1,x,tr] = 11
N1300 $P_UIFR[1,y,tr] = 14
N1310
N1320 g54
N1330
```

```
; Set frame
N1350 ROT RPL=-45
N1360 ATRANS X-2 Y10
N1370
```

```
; Four-edge roughing
N1390 G1 X10 Y-10 G41 OFFN=1; allowance 1 mm
N1400 X-10
N1410 Y10
N1420 X10
N1430 Y-10
N1440
```

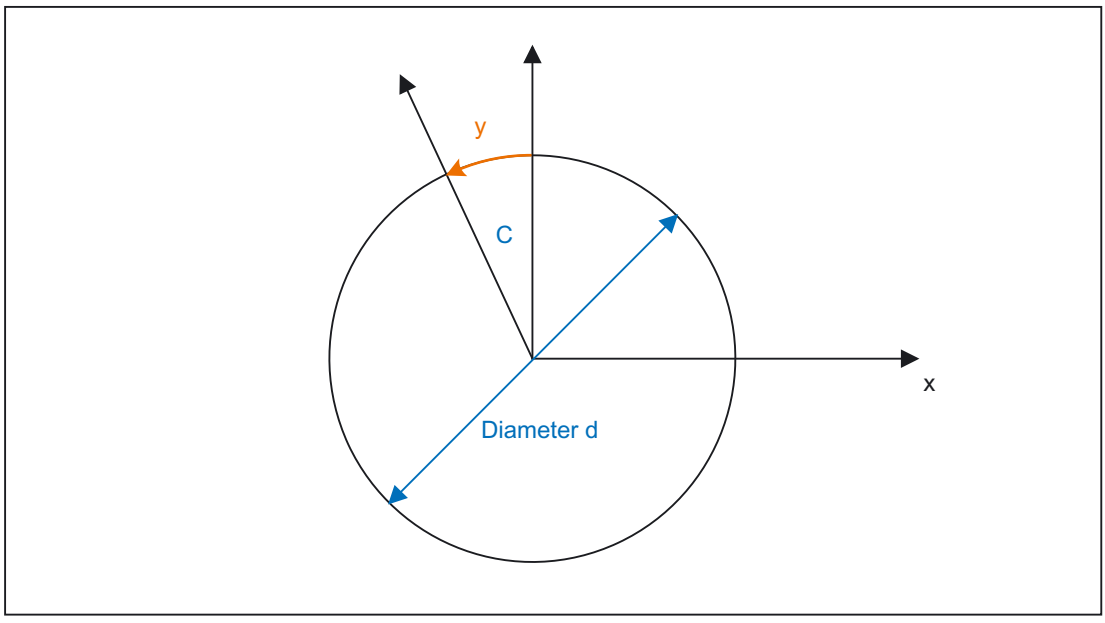
```
; Tool change
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490
```

```
; Four-edge finishing
N1510 G1 X10 Y-10
N1520 X-10
N1530 Y10
N1540 X10
N1550 Y-10
N1560
```

```
; Deselect frame
 N2950 m30 N1580 Z20 G40
N1590 TRANS
N1600
N1610 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,20,CAZ,30,C,15)
N1620 setal(61000)
N1630 endif
N1640 if $P_BFRAME <> $P_CHBFR[0]
N1650 setal(61000)
N1660 endif
N1670 if $P_IFRAME <>
TRANS(X,11,Y,0,Z,2,CAZ,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1680 setal(61000)
N1690 endif
N1730 if $P_ACTFRAME <>
TRANS(X,21,Y,0,Z,22,CAZ,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1740 setal(61001)
N1750 endif
N1760
N1770 TRAFOOF
N1780
N1790 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,C,15)
N1800 setal(61000)
N1810 endif
N1820 if $P_BFRAME <> $P_CHBFR[0]
N1830 setal(61000)
N1840 endif
N1850 if $P_IFRAME <>
TRANS(X,11,Y,2,Z,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1860 setal(61000)
N1870 endif
N1880 if $P_IFRAME <> $P_UIFR[1]
N1890 setal(61000)
N1900 endif
N1910 if $P_ACTFRAME <>
TRANS(X,21,Y,22,Z,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1920 setal(61002)
N1930 endif
N1940
N2010 $P_UIFR[1] = ctrans()
N2011 $P_CHBFR[0] = ctrans()
N2020 $P_UIFR[1] = ctrans(x,1,y,2,z,3,c,0)
N2021 G54
N2021 G0 X20 Y0 Z10 C0
```

```
N2030 TRANSMIT(1)
N2040 TRANS x10 y20 z30
N2041 ATRANS y200
N2050 G0 X20 Y0 Z10
N2051 if $P_IFRAME <> CTRANS(X,1,Y,0,Z,3,CAY,2)
N2052 setal(61000)
N2053 endif
N2054 if $P_ACTFRAME <> CTRANS(X,11,Y,20,Z,33,CAY,2):CFINE(Y,200)
N2055 setal(61002)
N2056 endif
N2060 TRAFOOF
N2061 if $P_IFRAME <> $P_UIFR[1]
N2062 setal(61000)
N2063 endif
N2064 if $P_ACTFRAME <> CTRANS(X,11,Y,2,Z,33):CFINE(Y,0)
N2065 setal(61002)
N2066 endif
```

TRACYL



TRACYL expansions:

The machine data below can be used to take the axial complete frame of the TRACYL rotary axis, i.e., the translation, fine offset, mirroring and scaling, into account in the transformation:

MD24805 $MC_TRACYL_ROT_AX_FRAME_1 = 1

MD24855 $MC_TRACYL_ROT_AX_FRAME_2 = 1

A rotary axis offset can, for example, be entered by compensating the oblique position of a workpiece in a frame within a frame chain. As a rule, this offset can also be included in the transformation as an offset in the rotary axis or as a y offset. A c axis offset, as in the figure above, then leads to corresponding x and y values.

MD24805 $MC_TRACYL_ROT_AX_FRAME_1 = 2

MD24855 $MC_TRACYL_ROT_AX_FRAME_2 = 2

With this setting, the axial offset of the rotary axis is taken account of in the transformation up to the SZS. The axial offsets of the rotary axis included in the SZS frames are entered into the transformation frame as offsets on the peripheral surface. This setting is only effective if the transformation frame has been configured.

Frame expansions:

The expansions described below are only valid for the following machine data settings:

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

The selection of transformation TRACYL produces a virtual geometry axis on the peripheral surface, coupled via the rotary axis, which is only taken into account in the contour frame but does not have a reference to an axial frame. All components of the virtual geometry axis are deleted. All other geometry axes accept their axial components when the transformation is selected.

Translations:

When selecting TRACYL, translations of the virtual axis are deleted. Translations of the rotary axis can be taken into account in the transformation.

Rotations:

Rotation before the transformation is taken over.

Mirroring:

Mirroring of the virtual axis is deleted. Mirroring of the rotary axis can be taken into account in the transformation.

Scaling:

Scaling of the virtual axis is deleted. Scaling of the rotary axis can be taken into account in the transformation.

Example:

Machine data for TRACYL:

```
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK = 'H41'          ; TRAFRAME, SETFRAME
$MC_CHSFRAME_RESET_MASK = 'H41'           ; Frames are active after Reset.
$MC_CHSFRAME_POWERON_MASK = 'H41'         ; Frames are deleted for Power On.

$MN_FRAME_GEOAX_CHANGE_MODE = 1           ; Frames are calculated after switchover
                                          of the geo axis.

$MC_RESET_MODE_MASK = 'H4041'             ; Basic frame is not deselected after
                                          Reset.
;$MC_RESET_MODE_MASK = 'H41'              ; Basic frame is deselected after Reset.

;$MC_GCODE_RESET_VALUES[7] = 2            ; G54 is the default setting.
$MC_GCODE_RESET_VALUES[7] = 1             ; G500 is the default setting.

$MN_MM_NUM_GLOBAL_USER_FRAMES = 0
$MN_MM_NUM_GLOBAL_BASE_FRAMES = 3

$MC_MM_NUM_USER_FRAMES = 10               ; from 5 to 100
$MC_MM_NUM_BASE_FRAMES = 3                ; from 0 to 8

$MN_NCBFRAME_RESET_MASK = 'HFF'
$MC_CHBFRAME_RESET_MASK = 'HFF'

$MN_MIRROR_REF_AX = 0                     ; No scaling when mirroring.
$MN_MIRROR_TOGGLE = 0
$MN_MM_FRAME_FINE_TRANS = 1               ; Fine offset
$MC_FRAME_ADD_COMPONENTS = TRUE           ; G58, G59 is possible
```

```
; TRACYL with groove side offset is 3rd transformer

$MC_TRAFO_TYPE_3 = 513; TRACYL

$MC_TRAFO_AXES_IN_3[0] = 1
$MC_TRAFO_AXES_IN_3[1] = 5
$MC_TRAFO_AXES_IN_3[2] = 3
$MC_TRAFO_AXES_IN_3[3] = 2

$MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] = 1
$MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] = 5
$MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] = 3

$MC_TRACYL_BASE_TOOL_1[0] = 0.0
$MC_TRACYL_BASE_TOOL_1[1] = 0.0
$MC_TRACYL_BASE_TOOL_1[2] = 0.0

$MC_TRACYL_ROT_AX_OFFSET_1 = 0.0
$MC_TRACYL_ROT_SIGN_IS_PLUS_1 = TRUE

$MC_TRACYL_ROT_AX_FRAME_1 = 1
```

Part program:

```
;Simple traversing test with groove side offset
N450 G603
N460
```

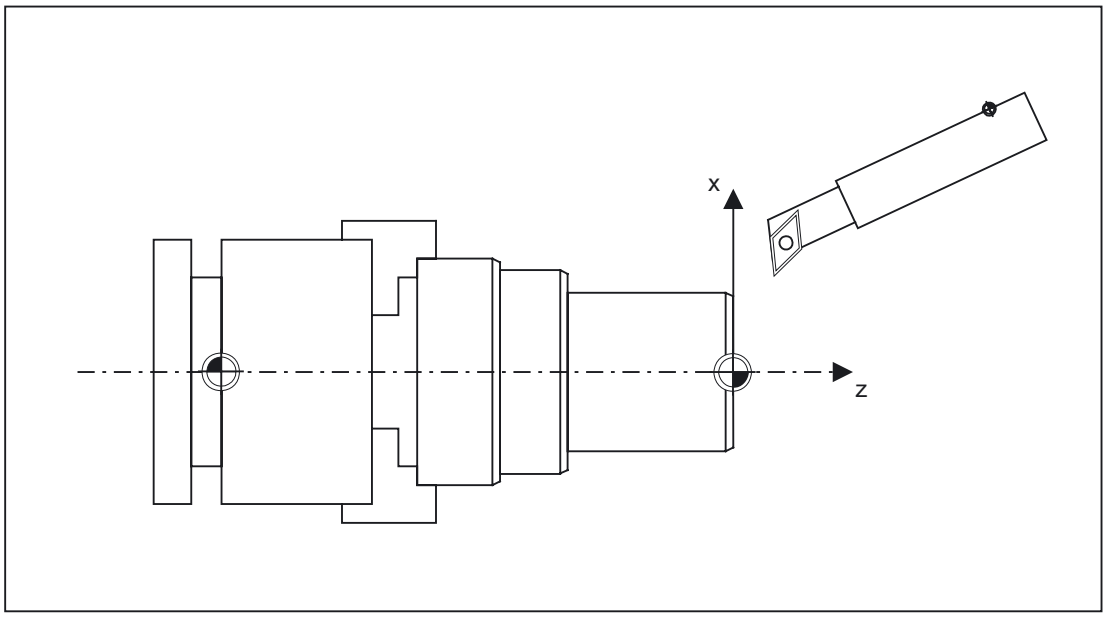
```
; Frame settings
N500 $P_UIFR[1] = ctrans(x,1,y,2,z,3,b,4)
N510 $P_UIFR[1] = $P_UIFR[1] : crot(x,10,y,20,z,30)
N520 $P_UIFR[1] = $P_UIFR[1] : cmirror(x,b)
N530
N540 $P_CHBFR[0] = ctrans(x,10,y,20,z,30,b,15)
N550
N560 G54
N570
```

```
; Continuous-path mode with selected smoothing
N590 G0 x0 y0 z-10 b0 G90 F50000 T1 D1 G19 G641 ADIS=1 ADISPOS=5
N600
N610 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,15)
N620 setal(61000)
N630 endif
N640 if $P_BFRAME <> $P_CHBFR[0]
N650 setal(61000)
N660 endif
N670 if $P_IFRAME <>
TRANS(X,1,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N680 setal(61000)
N690 endif
N700 if $P_IFRAME <> $P_UIFR[1]
N710 setal(61000)
N720 endif
N730 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N740 setal(61000)
N750 endif
N760
```

```
; Transformation ON
N780 TRACYL(40.)
N790
N800 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N810 setal(61000)
N820 endif
N830 if $P_CHBFR[0] <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N840 setal(61000)
N850 endif
N860 if $P_IFRAME <>
TRANS(X,1,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N870 setal(61000)
N880 endif
N890 if $P_UIFR[1] <>
TRANS(X,1,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N900 setal(61000)
N910 endif
N920 if $P_ACTFRAME <>
TRANS(X,11,Y,0,Z,33,CAY,22,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N930 setal(61001)
N940 endif
N950
N960 $P_UIFR[1,x,tr] = 11
N970 $P_UIFR[1,y,tr] = 14
N980
N990 g54
N1000
N1010 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N1020 setal(61000)
N1030 endif
N1040 if $P_BFRAME <> $P_CHBFR[0]
N1050 setal(61000)
N1060 endif
N1070 if $P_IFRAME <>
TRANS(X,11,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1080 setal(61000)
N1090 endif
N1100 if $P_IFRAME <> $P_UIFR[1]
N1110 setal(61000)
N1120 endif
N1130 if $P_ACTFRAME <>
TRANS(X,21,Y,0,Z,33,CAY,22,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1140 setal(61001)
N1150 endif
N1160
```

```
; Transformation off
N1180 TRAFOOF
N1190
N1200 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,15)
N1210 setal(61000)
N1220 endif
N1230 if $P_BFRAME <> $P_CHBFR[0]
N1240 setal(61000)
N1250 endif
N1260 if $P_IFRAME <>
TRANS(X,11,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1270 setal(61000)
N1280 endif
N1290 if $P_IFRAME <> $P_UIFR[1]
N1300 setal(61000)
N1310 endif
N1320 if $P_ACTFRAME <>
TRANS(X,21,Y,22,Z,33,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1330 setal(61002)
N1340 endif
N1350
N1360 G00 x0 y0 z0 G90
N1370
N1380 m30
```

TRAANG



Frame expansions:

The expansions described below are only valid for the following machine data settings:

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

Translations:

When selecting TRAANG, translations of the virtual axis are deleted.

Rotations:

Rotation before the transformation is taken over.

Mirroring:

Mirrorings of the virtual axis are taken over.

Scaling:

Scalings of the virtual axis are taken over.

Example:

Machine data for TRAANG:

```
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK = 'H1'            ; SETFRAME
$MC_CHSFRAME_RESET_MASK = 'H41'            ; Frames are active after RESET.
$MC_CHSFRAME_POWERON_MASK = 'H41'          ; Frames are deleted for "Power On".

$MN_FRAME_GEOAX_CHANGE_MODE = 1            ; Frames are calculated after switchover
                                           of the geo axis.

$MC_RESET_MODE_MASK = 'H4041'              ; Basic frame is not deselected after
                                           RESET.
;$MC_RESET_MODE_MASK = 'H41'               ; Basic frame is deselected after RESET.

;$MC_GCODE_RESET_VALUES[7] = 2             ; G54 is the default setting.
$MC_GCODE_RESET_VALUES[7] = 1              ; G500 is the default setting.

$MN_MM_NUM_GLOBAL_USER_FRAMES = 0
$MN_MM_NUM_GLOBAL_BASE_FRAMES = 3

$MC_MM_NUM_USER_FRAMES = 10                ; from 5 to 100
$MC_MM_NUM_BASE_FRAMES = 3                 ; from 0 to 8

$MN_NCBFRAME_RESET_MASK = 'HFF'
$MC_CHBFRAME_RESET_MASK = 'HFF'

$MN_MIRROR_REF_AX = 0                      ; No scaling when mirroring.
$MN_MIRROR_TOGGLE = 0
$MN_MM_FRAME_FINE_TRANS = 1                ; Fine offset
$MC_FRAME_ADD_COMPONENTS = TRUE            ; G58, G59 is possible.
```

```
; TRAANG is 1st transformer

$MC_TRAFO_TYPE_1 = 1024

$MC_TRAFO_AXES_IN_1[0] = 4                ; Inclined axis
$MC_TRAFO_AXES_IN_1[1] = 3                ; Axis is parallel to z
$MC_TRAFO_AXES_IN_1[2] = 2
$MC_TRAFO_AXES_IN_1[3] = 0
$MC_TRAFO_AXES_IN_1[4] = 0

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 3

$MC_TRAANG_ANGLE_1 = 85.
$MC_TRAANG_PARALLEL_VELO_RES_1 = 0.
$MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.

$MC_TRAANG_BASE_TOOL_1[0] = 0.0
$MC_TRAANG_BASE_TOOL_1[1] = 0.0
$MC_TRAANG_BASE_TOOL_1[2] = 0.0
```

```
; TRAANG is 2nd transformer

$MC_TRAFO_TYPE_2 = 1024

$MC_TRAFO_AXES_IN_2[0] = 4
$MC_TRAFO_AXES_IN_2[1] = 3
$MC_TRAFO_AXES_IN_2[2] = 0
$MC_TRAFO_AXES_IN_2[3] = 0
$MC_TRAFO_AXES_IN_2[4] = 0

$MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] = 4
$MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] = 0
$MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] = 3

$MC_TRAANG_ANGLE_2 = -85.
$MC_TRAANG_PARALLEL_VELO_RES_2 = 0.2
$MC_TRAANG_PARALLEL_ACCEL_RES_2 = 0.2

$MC_TRAANG_BASE_TOOL_2[0] = 0.0
$MC_TRAANG_BASE_TOOL_2[1] = 0.0
$MC_TRAANG_BASE_TOOL_2[2] = 0.0
```

Part program:

```
; Frame settings
N820 $P_UIFR[1] = ctrans(x,1,y,2,z,3,b,4,c,5)
N830 $P_UIFR[1] = $P_UIFR[1] : crot(x,10,y,20,z,30)
N840 $P_UIFR[1] = $P_UIFR[1] : cmirror(x,c)
N850
N860 $P_CHBFR[0] = ctrans(x,10,y,20,z,30,b,40,c,15)
N870
```

```
; Tool selection, clamping compensation, plane selection
N890 T2 D1 G54 G17 G90 F5000 G64 SOFT
N900
```

```
; Approach start position
N920 G0 X20 Z10
N930
N940 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,40,C,15)
N950 setal(61000)
N960 endif
N970 if $P_BFRAME <> $P_CHBFR[0]
N980 setal(61000)
N990 endif
N1000 if $P_IFRAME <>
TRANS(X,1,Y,2,Z,3,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1010 setal(61000)
N1020 endif
N1030 if $P_IFRAME <> $P_UIFR[1]
N1040 setal(61000)
N1050 endif
N1060 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1070 setal(61000)
N1080 endif
N1090
N1100 TRAANG(,1)
N1110
N1120 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,CAX,10,B,40,C,15)
N1130 setal(61000)
N1140 endif
N1150 if $P_BFRAME <> $P_CHBFR[0]
N1160 setal(61000)
```

```
N1170 endif
N1180 if $P_IFRAME <>
CTRANS(X,1,Y,2,Z,3,CAX,1,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1190 setal(61000)
N1200 endif
N1210 if $P_IFRAME <> $P_UIFR[1]
N1220 setal(61000)
N1230 endif
N1240 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,CAX,11,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1250 setal(61001)
N1260 endif
N1270
N1280
N1290 $P_UIFR[1,x,tr] = 11
N1300 $P_UIFR[1,y,tr] = 14
N1310
N1320 g54
N1330
```

```
; Set frame
N1350 ROT RPL=-45
N1360 ATRANS X-2 Y10
N1370
```

```
; Four-edge roughing
N1390 G1 X10 Y-10 G41 OFFN=1; allowance 1 mm
N1400 X-10
N1410 Y10
N1420 X10
N1430 Y-10
N1440
```

```
; Tool change
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490
```

```
; Four-edge finishing
N1510 G1 X10 Y-10
N1520 X-10
N1530 Y10
N1540 X10
N1550 Y-10
N1560
```

```
; Deselect frame
N1580 Z20 G40
N1590 TRANS
N1600
N1610 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,CAX,10,B,40,C,15)
N1620 setal(61000)
N1630 endif
N1640 if $P_BFRAME <> $P_CHBFR[0]
N1650 setal(61000)
N1660 endif
N1670 if $P_IFRAME <>
TRANS(X,11,Y,14,Z,3,CAX,1,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1680 setal(61000)
N1690 endif
N1700 if $P_IFRAME <> $P_UIFR[1]
N1710 setal(61000)
N1720 endif
N1730 if $P_ACTFRAME <>
TRANS(X,21,Y,34,Z,33,CAX,11,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1740 setal(61001)
N1750 endif
N1760
N1770 TRAFOOF
N1780
N1790 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,40,C,15)
N1800 setal(61000)
N1810 endif
N1820 if $P_BFRAME <> $P_CHBFR[0]
N1830 setal(61000)
N1840 endif
N1850 if $P_IFRAME <>
TRANS(X,1,Y,14,Z,3,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1860 setal(61000)
N1870 endif
N1880 if $P_IFRAME <> $P_UIFR[1]
N1890 setal(61000)
N1900 endif
N1910 if $P_ACTFRAME <>
TRANS(X,11,Y,34,Z,33,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1920 setal(61002)
N1930 endif
N1940
N1950 m30
```

9.5.6.3 Adapting active frames

The geometry axis configuration can change during program execution or on RESET. The number of available geometry axes can vary from zero to three. With unavailable geometry axes, components in the active frames (e.g., rotations) can lead to the active frames for this axis configuration becoming invalid. This is indicated by the alarms below:

Channel %1 block %2 rotation programmed for non-existent geometry axis

This alarm remains present until the frames have been changed accordingly.

The machine data below can be used to switch on the automatic adaptation of active frames, thus preventing alarm 16440:

MD24040 $MC_FRAME_ADAPT_MODE

Bitmask for adapting the active frames with reference to the axis constellation.

The following settings apply:

Bit 0: Rotations in active frames, which rotate coordinate axes with no geometry axes, are deleted from the active frames.

Bit 1: Shear angles in the active frames are orthogonalized.

Bit 2: Scalings of all geometry axes in the active frames are set to value 1.

With machine data setting
MD24040 $MC_FRAME_ADAPT_MODE = 1
, all rotations in the active frames, which could produce coordinate axis movements for non-existent geometry axes, are deleted.
The data management frames are not changed. Only the possible rotations are applied when the data management frames are activated.

Example:

No y axis exists:

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

$P_UIFR[1] = crot(x,45,y,45,z,45)

```
N390 G54 G0 X10 z10 f10000

if $P_IFRAME <> crot(y,45)                ; Only the rotation about y is taken
                                          over.
setal(61000)
endif
```

9.5.6.4 Mapped Frames

Overview

The "mapped frames" function supports the cross-channel consistent change of axial frames inside channel-specific or global data management frames. Using axial machine data, it is defined between which axes mapping is realized.

If frame mapping is, e.g. active for machine axes AX1 and AX4, and the axial frame of axis AX1 is changed in a channel-specific data management frame (e.g. basic frame $P_CHBFR[x]) (translation, fine translation, scaling, mirroring), then this frame data for AX1 and AX4 is transferred to all channel-specific data management frames (e.g. basic frame $P_CHBFR[x]) in all channels in which they are parameterized as channel axes.

Frame mapping is not realized when changing the axial frame data for the rotation.

Requirements

The following requirements must be fulfilled for frame mapping:

- The data management frames used for mapping must be configured:
  MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK (system frames)
- Channel-specific data management frames must be explicitly enabled for mapping:
  MD10616 $MN_MAPPED_FRAME_MASK (enable frame mapping)

  Note
  For global data management frames, mapping is always carried out. An enable is not required.

Parameterization

The parameterization of the mapping relationships is realized in the axis-specific machine data:

MD32075 $MA_MAPPED_FRAME[<AXn>] = "AXm"

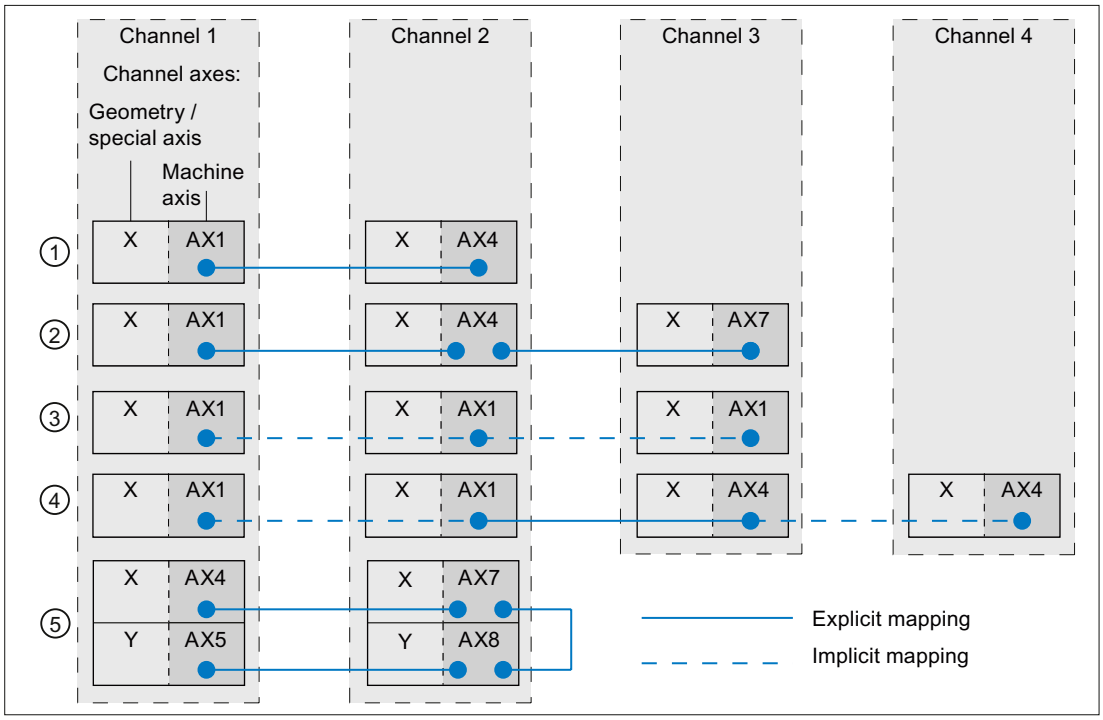
AXn, AXm: Machine axis identifier with n, m = 1, 2, ... max. number of machine axes

Mapping rules

The following rules apply for frame mapping:

- The mapping is bidirectional.
  An axial frame can be written for AXn or AXm. The frame data is always accepted and taken for the other axis.
- All parameterized mapping relationships are always evaluated.
  When writing an axial frame of axes AXn, all mapping relationships are evaluated and the frame data accepted for all directly and indirectly involved axes.
- The mapping is global across all channels.
  When writing an axial frame of axis AXn or AXm for a channel-specific frame, the frame data is accepted for all channels in which AXn or AXm are parameterized as channel axes.
- When writing an axial frame using geometry or special axis identifier, the mapping relationships are evaluated via the machine axes currently assigned to the geometry or special axis.
- Mapping is frame-specific.
  When writing an axial frame, the frame data is only mapped within the same channel-specific or global data management frame.

NOTICE
Data consistency The user / machine manufacturer is solely responsible for ensuring that after a frame is written, consistent frame data is available in all channels, e.g. by using channel synchronization.



Description	Parameterization: $MA_
① Simple mapping relationship: AX1(K1) ↔ AX4(K2)	MAPPED_FRAME[<AX1>] = "AX4"
② Chained mapping relationships: AX1(K1) ↔ AX4(K2) ↔ AX7(K3)	MAPPED_FRAME[<AX1>] = "AX4" MAPPED_FRAME[<AX4>] = "AX7"
③ Mapping relationship to itself, with AX1 as channel axis of channels 1, 2 and 3: AX1(K1+K2+K3)	MAPPED_FRAME[<AX1>] = "AX1"
④ Mapping relationship between two axes, the channel axes in two channels are: AX1(K1+K2) ↔ AX4(K3+K4)	MAPPED_FRAME[<AX1>] = "AX4"
⑤ Chained mapping relationships where multiple channel axes can be written in the same channel: AX4(K1) ↔ AX7(K2) ↔ AX8(K2) ↔ AX5(K1)	MAPPED_FRAME[<AX4>] = "AX7" MAPPED_FRAME[<AX7>] = "AX8" MAPPED_FRAME[<AX8>] = "AX5"

Figure 9-23 Mapping examples

Activating the data management frame

Data management frames can be written in the part program and via the user interface of SINUMERIK Operate. The following should be noted when activating the data management frames in the channels written directly and via frame mapping:

- Writing in the part program
  The data management frames must be explicitly activated in each channel (G500, G54 ... G599)
- Writing via user interface
  Data management frames are written via the user interface, e.g. by entering new zero offsets. A modified data management frame is immediately active in all of the involved channels if none of these channels is in the "Channel active" state. The data management frame is not active in any channel if one of the channels involved is in the "Channel active" state. The activation must then be explicitly programmed in each channel in the part program (G500, G54 ... G599). Or, the next time that the channel state changes, it becomes active after "Channel reset".

Example

The following channels and channel axes are parameterized at a control:

- Channel 1
  - Z: Geometry axis
  - AX1: Machine axis of geometry axis Z
- Channel 2
  - Z: Geometry axis
  - AX4: Machine axis of geometry axis Z

The zero point of the Z axis should always be the same in both channels:

- Mapping relationship: $MA_MAPPED_FRAME[AX1] = "AX4"

Programming in the part program

```
Channel 1                                   Channel 2
...                                         ...
N100 WAIT (10,1,2)                          N200 WAIT (10,1,2)
N110 $P_UIFR[1] = CTRANS(Z, 10)
N120 WAIT (20,1,2)                          N220 WAIT (20,1,2)
N130 G54                                    N230 G54
N140 IF ($P_IFRAME[0, Z, TR] <> 10)         N230 IF ($P_IFRAME[0, Z, TR] <> 10)
N150   SETAL(61000)                         N250   SETAL(61000)
N160 ENDIF                                  N260 ENDIF
...                                         ...
```

Description:

N100 / N200	Channel synchronization for consistent writing and mapping of frame data
N110	Writing of the settable data management frame $P_UIFR[1]: Moving the zero point of the Z axis to 10 mm Mapping the axial frame data: Channel 1: Z ≙ AX1 ⇔ channel 2: Z ≙ AX4
N120 / N220	Channel synchronization for consistent activation of new frame data
N130 / N230	Activating the new frame data
N140 / N240	Checking the zero point of the Z axis for: 10 mm

9.5.7 Predefined frame functions

9.5.7.1 Inverse frame

To round off the frame arithmetic, the part program provides a function which calculates the inverse frame from another frame. The chaining between a frame and its inverse frame always produces a zero frame.

FRAME INVFRAME(FRAME)

Frame inversion is an aid for coordinate transformations. Measuring frames are usually calculated in the WCS. If you should wish to transform this calculated frame into another coordinate system, i.e., the calculated frame should be entered into any desired frame within the frame chain, the calculations below should be used.

The new complete frame is a chain of the old complete frame and the calculated frame.

$P_ACTFRAME = $P_ACTFRAME : $AC_MEAS_FRAME

The new frame in the frame chain is therefore:

Target frame is $P_SETFRAME:

$P_SETFRAME = $P_ACTFRAME : $AC_MEAS_FRAME : INVFRAME($P_ACTFRAME) : $P_SETFRAME

Target frame is nth channel basic frame $P_CHBFRAME[n]:

n = 0: TMP = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_NCBFRAME[0..k]

n <> 0: TMP = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_NCBFRAME[0..k] : $P_CHBFRAME[0..n-1]

k = $MN_MM_NUM_GLOBAL_BASE_FRAMES

$P_CHBFRAME[n] = INVFRAME(TMP) : $P_ACTFRAME : $AC_MEAS_FRAME : INVFRAME($P_ACTFRAME) : TMP : $P_CHBFRAME[n]

Target frame is $P_IFRAME:

TMP = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_BFRAME

$P_IFRAME = INVFRAME(TMP) : $P_ACTFRAME : $AC_MEAS_FRAME : INVFRAME($P_ACTFRAME) : TMP : $P_IFRAME

Application example:

A frame calculated, for example, via a measuring function, must be entered in the current SETFRAME such that the new complete frame is a chain of the old complete frame and the measurement frame. The SETFRAME is calculated accordingly by means of frame inversions.

```
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

$P_CHBFRAME[0] = crot(z,45)
```

```
$P_IFRAME[x,tr] = -sin(45)
$P_IFRAME[y,tr] = -sin(45)

$P_PFRAME[z,rt] = -45

; Measure corner with four measuring points
$AC_MEAS_VALID = 0

; Approach measuring point 1
g1 x-1 y-3

; Store measuring point 1
$AC_MEAS_LATCH[0] = 1

; Approach measuring point 2
g1 x5 y-3

; Store measuring point 2
$AC_MEAS_LATCH[1] = 1

; Approach measuring point 3
g1 x-4 y4

; Store measuring point 3
$AC_MEAS_LATCH[2] = 1

; Approach measuring point 4
g1 x-4 y1

; Store measuring point 4
$AC_MEAS_LATCH[3] = 1
```

```
; Set position setpoint of the corner
$AA_MEAS_SETPOINT[x] = 0
$AA_MEAS_SETPOINT[y] = 0
$AA_MEAS_SETPOINT[z] = 0

; Define setpoint angle of intersection
$AC_MEAS_CORNER_SETANGLE = 90
$AC_MEAS_WP_SETANGLE = 30

; Measuring plane is G17
$AC_MEAS_ACT_PLANE = 0

;Select tool
$AC_MEAS_T_NUMBER = 1
$AC_MEAS_D_NUMBER = 1

; Set measuring type on corner 1
$AC_MEAS_TYPE = 4

; Start measuring process
RETVAL = MEASURE()

if RETVAL <> 0
setal(61000 + RETVAL)
endif

if $AC_MEAS_WP_ANGLE <> 30
setal(61000 + $AC_MEAS_WP_ANGLE)
endif
```

```
if $AC_MEAS_CORNER_ANGLE <> 90
setal(61000 + $AC_MEAS_CORNER_ANGLE)
endif

; Transform measured frame and write in accordance with $P_SETFRAME in such a way
; that a complete frame is produced, as a result of the old complete frame
; being chained with the measuring frame.

$P_SETFRAME = $P_ACTFRAME : $AC_MEAS_FRAME : INVFRAME($P_ACTFRAME) :
$P_SETFRAME

; Describe system frames in data management
$P_SETFR = $P_SETFRAME

; Approach the corner
g1 x0 y0

; Retract the rectangle rotated about 30 degrees
g1 x10
y10
x0
y0

m30
```

9.5.7.2 Additive frame in frame chain

Measurements on the workpiece or calculations in the part program and cycles generally produce a frame that is applied additively to the current complete frame. The WCS and thus the programming zero must, therefore, be displaced and possibly rotated. This measured frame is available as a temporary frame and not yet actively included in the frame chain. This function is used to calculate and possible activate this frame:

```
INT ADDFRAME(FRAME,STRING)
```

Programming

Parameter 1:	Type: FRAME	Additively measured or calculated frame
Parameter 2:	Type: STRING	Strings for current frames: "$P_CYCFRAME", "$P_ISO4FRAME", "$P_PFRAME", "$P_WPFRAME", "$P_TOOLFRAME", "$P_IFRAME", "$P_CHBFRAME[0..16]", "$P_NCBFRAME[0..16]", "$P_ISO1FRAME", "$P_ISO2FRAME", "$P_ISO3FRAME", "$P_EXTFRAME", "$P_SETFRAME" "$P_PARTFRAME" Strings for data management frames: "$P_CYCFR", "$P_ISO4FR, "$P_TRAFR", "$P_WPFR", "$P_TOOLFR", "$P_UIFR[0..99]", "$P_CHBFR[0..16]", "$P_NCBFR[0..16]", "$P_ISO1FR, "$P_ISO2FR, "$P_ISO3FR, "$P_EXTFR", "$P_SETFR", "$P_PARTFR"

Function value:	Type: INT	Value:	Significance:
		0	OK
		1	Specified target (string) is wrong
		2	Target frame is not configured
		3	Rotation in frame is not permitted

The ADDFRAME() function calculates the target frame, which is specified by the STRING. The target frame is calculated in such a way that the new complete frame appears as a chain of the old complete frame and the transferred frame, e.g.:

```
ERG = ADDFRAME(TMPFRAME,"$P_SETFRAME")
```

The new complete frame is calculated to be:

$P_ACTFRAME$_{new}$ = $P_ACTFRAME$_{old}$: TMPFRAME

If a current frame has been specified as a target frame, then the new complete frame becomes active at the preprocessing stage. If the target frame is a data management frame, then the frame is not operative until it is explicitly activated in the part program.

The function does not set any alarms, but returns the error codes via the return value. The cycle can react according to the error codes.

9.5.8 Functions

9.5.8.1 Setting zeros, workpiece measuring and tool measuring

PRESET is achieved using HMI operator actions or measuring cycles. The calculated frame can be written to system frame SETFRAME. The position setpoint of an axis in the WCS can be altered when the actual-value memory is preset.

"Scratching" means workpiece and tool 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 or the hole, position setpoints can be added to the measured positions in the WCS. The resultant offsets can be entered in a selected frame. In tool measuring, the length or radius of a tool can be measured using a measured reference part.

Measurements can be taken via operator actions or measuring cycles. Communication with the NCK takes place via predefined system variables. In the NCK, the calculation is made by using a HMI operator action to activate a PI service, or by using a part-program command from the measuring cycles. A tool and a plane can be selected as a basis for the calculation. The calculated frame is entered in the result frame.

9.5.8.2 Zero offset external via system frames

Zero offset via PLC or part program

The amount of the external zero offset can be specified through HMI and PLC via BTSS or programmed in the part program via the axis-specific system variable $AA_ETRANS [<Axis>] = <Value>.

Activation
The activation of the external zero offset takes place through the interface signal:
DB31, ... DBX3.0 (accept external zero offset)

Behavior
After the activation the respective specified axis-specific amount of the external zero offset is traversed outside for each axis with the next possible traversing block.

Next possible means that enough dynamic reserves must be available for the respective axis for traversing the zero offset. If the axis is traversed in the next traversing block after activation owing to its programming with the maximum speed, dynamic reserve is no longer available for traversing the external zero offset.

Together with the continuous path mode G64 the traversing of the offset can stretch over several part program blocks.

Zero offset via system frame

The external zero offset can then also be managed and activated by the functionality described above via the system frame $P_EXTFRAME .

configuring

The configuring of the external zero offset is done via the system frame $P_EXTFRAME via bit1 = 1 in the machine data: MD28082 $MC_MM_SYSTEM_FRAME_MASK = 'B0010'
The amount for the external zero offset can be specified manually via the HMI user interface and the PLC user program via BTSS or programmed in the part program via the axial system variable $AA_ETRANS[<Axis>].

Activation

The activation of the external zero offset takes place through the interface signal:
DB31, ... DBX3.0 (accept external zero offset)

Behavior

Upon activation of the external zero offset the traversing movements of all axes, except command and PLC axes, are stopped immediately and the advance is reorganized. The rough offset of the current system frame and of the system frame in data management is set to the value of the axial system variable $AA_ETRANS[<axis>]. Thereafter, the offset is traversed first and then the interrupted movement is continued.

Behavior for incremental dimension

In case of active incremental dimension `G91` and machine data:
MD42440 $MC_FRAME_OFFSET_INCR_PROG (zero offset in frames) = 0
traversing the offset is done in the scope of the external zero offset via system frame, despite opposite configuring of the machine data, with the approach block, although it is specified by a frame.

Note

The external zero offset always acts absolutely.

9.5.8.3 Toolholder

Translations

With kinematics of type "P" and "M", the selection of a toolholder activates an additive frame (table offset of the orientable toolholder) which takes into account the zero offset as a result of the rotation of the table. The zero offset is entered in the system frame $P_PARTFR. In this case the translatory component of this frame is overwritten. The other frame components are retained.

The system frame $P_PARTFR must be enabled via the following machine data:
MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 2 = 1 (system frame for TCARR and PAROT)

Note

Alternatively, the offset can also parameterized via machine data to record the the table offset:

MD20184 $MC_TOCARR_BASE_FRAME_NUMBER = <number of the basic frame>

This option is only for compatibility reasons to older software versions. You are strongly recommended **not** to use this procedure any longer.

A frame offset as a result of a toolholder change becomes effective immediately on selection of TCARR=... A change in the tool length, on the other hand, only becomes effective immediately if a tool is active.

A frame rotation does not take place on activation, and a rotation which is already active is not changed. As in case T (only the tool can be rotated), the position of the rotary axes used for the calculation is dependent on the command TCOFR / TCOABS and determined from the rotation component of an active frame or from the entries $TC_CARRn. Activation of a frame changes the position in the workpiece coordinate system accordingly, without compensating motion by the machine itself.

The ratios are shown in the figure below:

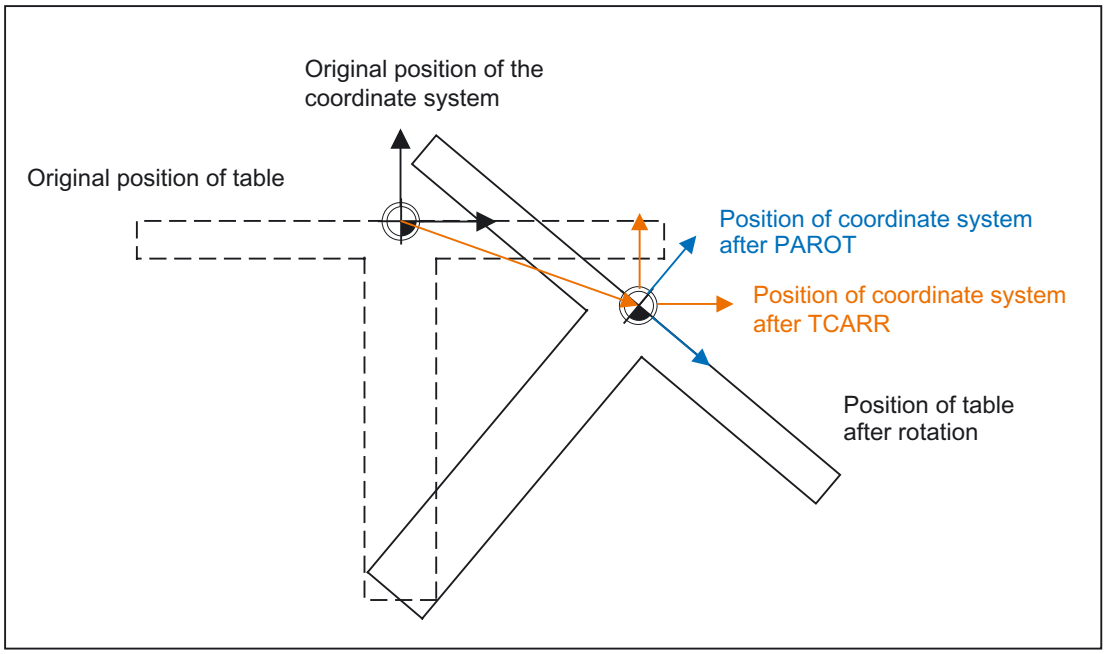


Figure 9-24 Frame on activation of a rotary table with TCARR

With kinematics of type M (tool and table are each rotary around one axis), the activation of a toolholder with TCARR simultaneously produces a corresponding change in the effective tool length (if a tool is active) and the zero offset.

Rotations

Depending on the machining task, it is necessary to take into account not only a zero offset (whether as frame or as tool length) when using a rotary toolholder or table, but also a rotation. However, the activation of a toolholder with orientation capability never leads directly to a rotation of the coordinate system.

If only the tool can be rotated, a frame can be defined for it using TOFRAME or TOROT.

With rotary tables (kinematics types P and M), activation with TCARR similarly does not lead to an immediate rotation of the coordinate system, i.e. even though the zero point of the coordinate system is offset relative to the machine, while remaining fixed relative to the zero point of the workpiece, the orientation remains unchanged in space.

If the coordinate system needs to be fixed relative to the workpiece, i.e. not only offset relative to the original position but also rotated according to the rotation of the table, then PAROT can be used to activate such a rotation in a similar manner to the situation with a rotary tool.

With PAROT, the translations, scalings and mirroring in the active frame are retained, but the rotation component is rotated by the rotation component of an orientational toolholder corresponding to the table. The entire programmable frame including its rotation component remains unchanged.

The rotation component that describes the rotation of the table is then either entered in the system frame $PARTFR or in the basic frame parameterized with MD20184 $MC_TOCARR_BASE_FRAME_NUMBER:

$MC_MM_SYSTEM_FRAME_MASK, bit 2 = <value>

Value	Meaning
1	Rotation component → $PARTFR
0	Rotation component → MD20184 $MC_TOCARR_BASE_FRAME_NUMBER

As with the note made in the description of the table offset, the second alternative here is not recommended for use with new systems.

The rotation component of the part frame can be deleted with PAROTOF, irrespective of whether this frame is in a basic or a system frame.
The translation component is deleted when a toolholder which does not produce an offset is activated or a possibly active orientable toolholder is deselected with TCARR=0.

PAROT or TOROT take into account the overall orientation change in cases where the table or the tool are oriented with two rotary axes. With mixed kinematics only the corresponding component caused by a rotary axis is considered. It is thus possible, for example, when using TOROT, to rotate a workpiece such that an inclined plane lies parallel to the XY plane fixed in space, whereby rotation of the tool must be taken into account in machining where any holes to be drilled, for example, are not perpendicular to this plane.

Example

On a machine, the rotary axis of the table points in the positive Y direction. The table is rotated by +45 degrees. PAROT defines a frame which similarly describes a rotation of 45 degrees around the Y axis. The coordinate system is not rotated relative to the actual environment (marked in the figure with "Position of the coordinate system after TCARR"), but is rotated by -45 degrees relative to the defined coordinate system (position after PAROT). If this coordinate system is defined with ROT Y-45, for example, and if the toolholder is then selected with active TCOFR, an angle of +45 degrees will be determined for the rotary axis of the toolholder.

Language command PAROT is not rejected if no toolholder with orientation capability is active. However, such a call then produces no frame changes.

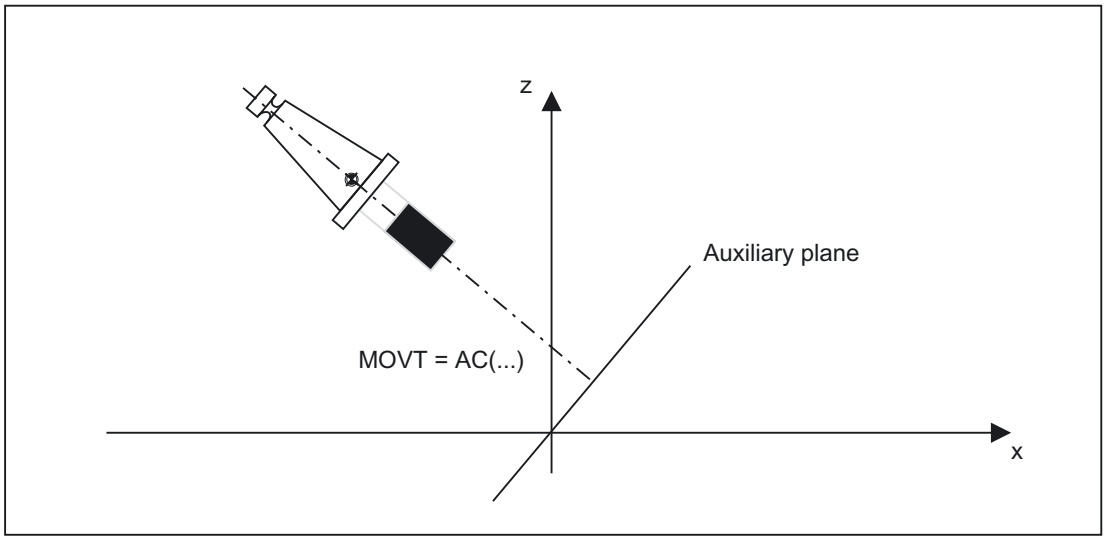
Machining in direction of tool orientation

Particularly on machines with tools that can be oriented, traversing should take place in the tool direction (typically, when drilling) without activating a frame (e.g. using TOFRAME or TOROT), on which one of the axes points in the direction of the tool. This is also a problem if, when carrying out inclined machining operations, a frame defining the inclined plane is active, but the tool cannot be set exactly perpendicularly because an indexed toolholder (Hirth tooth system) prevents free setting of the tool orientation. In these cases it is then necessary - contrary to the motion actually requested perpendicular to the plane - to drill in the tool direction, as the drill would otherwise not be guided in the direction of its longitudinal axis (tool breaks).

The end point of such a motion is programmed with MOVT= ....
The programmed value is effective incrementally in the tool direction as standard.
The positive direction is defined from the tool tip to the tool adapter. The content of MOVT is thus generally negative for the infeed motion (when drilling), and positive for the retraction motion. This corresponds to the situation with normal paraxial machining, e.g. with G91 Z ....

Instead of MOVT= ... it is also possible to write MOVT=IC(...) if it is to be plainly visible that MOVT is to function incrementally. There is no functional difference between the two forms.

If the motion is programmed in the form MOVT=AC(...), MOVT functions absolutely.
In this case a plane is defined which runs through the current zero point, and whose surface normal vector is parallel to the tool orientation. MOVT then gives the position relative to this plane (see figure). The reference plane is only used to calculate the end position. Active frames are not affected by this internal calculation.



Programming with MOVT is independent of the existence of a toolholder that can be oriented. The direction of the motion is dependent on the active plane.
It runs in the directions of the vertical axes, i.e. with G17 in Z direction, with G18 in Y direction and with G19 in X direction. This applies when no orientable toolholder is active and for the case of an orientable toolholder without rotary tool or with a rotary tool in its basic setting.

MOVT acts similarly for active orientation transformation (3-, 4-, 5-axis transformation).

If in a block with MOVT the tool orientation is changed simultaneously (e.g. active 5-axis transformation by means of simultaneous interpolation of the rotary axes), the orientation at the start of the block is decisive for the direction of motion of MOVT.
With an active 5-axis transformation, the path of the tool center point (TCP) is not affected by the change of orientation, i.e. the path remains a straight line and its direction is determined by the tool orientation at the start of the block.

If MOVT is programmed, linear or spline interpolation must be active (G0,G1, ASPLINE, BSPLINE, CSPLINE). Otherwise, an alarm is produced.
If a spline interpolation is active, the resultant path is generally not a straight line, since the end point calculated by MOVT is treated as if it had been programmed explicitly with X, Y, Z.

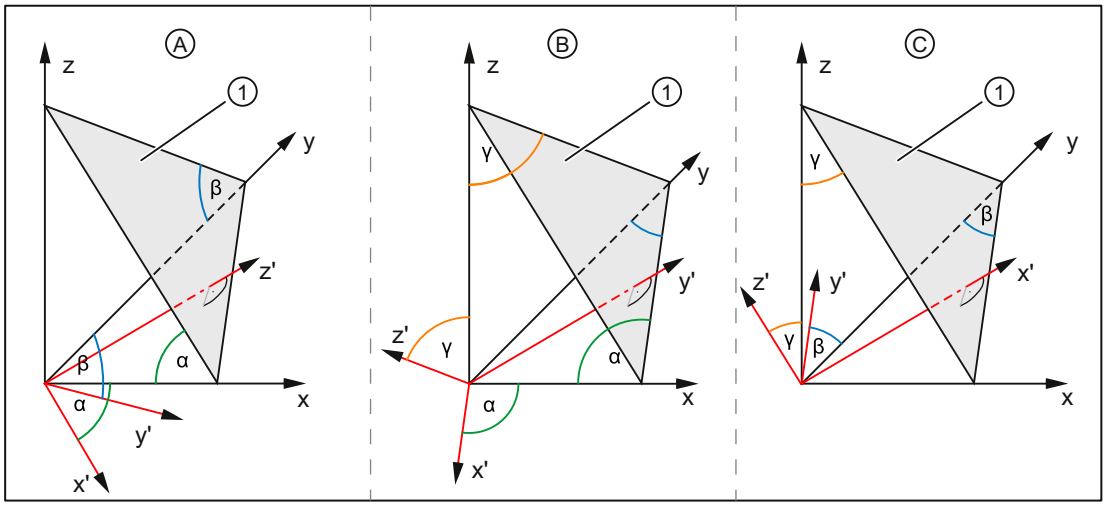
A block with MOVT must not contain any programming of geometry axes (alarm 14157).

Definition of frame rotations with solid angles

Where a frame is to be defined to describe a rotation around more than one axis, this is achieved through chaining individual rotations. A new rotation is hereby always performed in the already rotated coordinate system.
This applies both to programing in one block, e.g. with ROT X... Y... Z..., and when constructing a frame in several blocks, e.g. in the form:

```
N10 ROT Y...
N20 AROT X...
N30 AROT Z...
```

In workpiece drawings, inclined surfaces are frequently described by way of solid angles, i.e. the angles which the intersection lines of the inclined plane form with the main planes (X-Y, Y-Z, Z-X planes) (see figure). The machine operator is not expected to convert these solid angles into the angles of rotation of a chaining of individual rotations.



① Inclined plane

Figure 9-25 Rotation with solid angles

For this reason, the language commands ROTS, AROTS and CROTS are used, with which the rotations can be immediately described as solid angles.

The orientation of a plane in space is defined unambiguously by specifying two solid angles. The third solid angle is derived from the first two. Therefore, a maximum of two solid angles may be programmed, e.g. in the form ROTS X10 Y15. If a third solid angle is specified, an alarm will be triggered.

It is permissible to specify a single solid angle. The rotations which are performed with ROTS or AROTS in this case are identical to those for ROT and AROT.

An expansion of the existing functionality arises only in cases where exactly two solid angles are programmed.

The two programmed axes define a plane, the non-programmed axis defines the related third axis of a right-hand coordinate system. Which axis is first and which second is then unambiguously defined for both programmed axes (the definition corresponds to those found in the plane definition of G17/G18/G19). The angle programmed with the axis letter of an axis of the plane then specifies the axis, around which the other axis of the plane must be rotated in order to move this into the line of intersection, which the rotated plane forms with the plane surrounded by the other and the third axis. This definition ensures that, in the case that one of the two programmed angles is towards zero, the defined plane enters the plane, which is created if only one axis is programmed (e.g. with ROT or AROT).

The diagram shows an example where X and Y are programmed. Y specifies the angle here, through which the X axis must rotate around the Y axis to bring the X axis to the line of intersection formed by the inclined plane and the X-Z plane. The same principle applies for the programmed value of X.

Note

In the shown position of the inclined plane the value of Y is positive, that of X on the other hand negative.

The specification of the solid angle does not define the orientation of the two-dimensional coordinate system within the plane (i.e. the angle of rotation around the surface normal vector). The position of the coordinate system is thus determined so that the rotated first axis lies in the plane which is formed by the first and third axes of the non-rotated coordinate system.

This means that

- When programming X and Y the new X axis lies in the old Z-X plane.
- When programming Z and X the new Z axis lies in the old Y-Z plane.
- When programming Y and Z the new Y axis lies in the old X-Y plane.

If the required coordinate system does not correspond to this basic setting, then an additional rotation must be performed with AROT....

The programmed solid angles are converted to the equivalent Euler angles according to the zy'x" convention (RPY angle) or zx'z" convention when entering depending on the machine data:
MD10600 $MN_FRAME_ANGLE_INPUT_MODE.
These then appear in the display.

Frame rotation in tool direction

With the language command TOFRAME, which also existed in older software versions, it is possible to define a frame whose Z axis points in the tool direction.
An already programmed frame is then overwritten by a frame which describes a pure rotation. Any zero offsets, mirrorings or scalings existing in the previously active frame are deleted.
This response is sometimes interfering. It is often particularly practical to retain a zero offset, with which the reference point in the workpiece is defined.

The language command TOROT is then also used. This command overwrites only the rotation component in the programmed frame and leaves the remaining components unchanged.
The rotation defined with TOROT is the same as that defined with TOFRAME.
TOROT is, like TOFRAME, independent of the availability of an orientational toolholder. This language command is also especially useful for 5-axis transformations.

The new language command TOROT ensures consistent programming with active orientational toolholders for each kinematics type.

TOFRAME or TOROT defines frames whose Z direction points in the tool direction. This definition is suitable for milling, where G17 is usually active. However, particularly with turning or, more generally, when G18 or G19 is active, it is desirable that frames which will be aligned on the X or Y axis, can be defined. For this purpose, the following commands are available in G group 53:

- TOFRAMEX, TOFRAMEY, TOFRAMEZ
- TOROTX, TOROTY, TOROTZ

The appropriate frames can be defined with these commands. The functions of TOFRAME and TOFRAMEZ or of TOROT and TOROTZ are identical to one another.

The frames resulting from TOROT or TOFRAME can be written in a separate system frame ($P_TOOLFR). The programmable frame is then retained unchanged.

- Requirement: MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 3 = 1

When programming TOROT or TOFRAME, etc. response is identical, with or without a system frame. Differences occur when the programmable frame is processed further elsewhere.

Note

In new systems, it is recommended that only the intended system frame is used for frames produced by the commands of G group 53.

Example

TRANS is programmed after TOROT. TRANS without specified parameters deletes the programmable frame. In the variant without a system frame, this also deletes the frame component of the programmable frame produced by TOROT, but if the TOROT component is in the system frame, it is retained.

TOROT or TOFRAME, etc. are disabled with language command TOROTOF. TOROTOF deletes the entire system frame $P_TOOLFR. If the programmable frame (old variant) and not the system frame is described by commands TOFRAME, etc. TOROT only deletes the rotation component and leaves the remaining frame components unchanged.

If a rotating frame is already active before language command TOFRAME or TOROT is activated, a request is often made that the newly defined frame should deviate as little as possible from the old frame. This is the case, for example, if a frame definition needs to be modified slightly because the tool orientation cannot be set freely on account of Hirth-toothed rotary axes. The language commands uniquely define the Z direction of the new frame.

Frame definition for TOFRAME, TOROT and PAROT

The following machine data can be used to select one of two variants for the position of the X axis and Y axis. However, in both cases there is no reference to the previously active frame.

MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE (coordinate system for automatic frame definition)

For this reason, it is recommended that the following setting data be used instead so that the behavior of TOFRAME and TOROT can be specifically controlled.

SD42980 $SC_TOFRAME_MODE (frame definition for TOFRAME, TOROT and PAROT)

Value	Meaning
1	The new X direction is chosen to lie in the X-Z plane of the old coordinate system. In this setting, the angle difference between the old and new Y axis will be minimal.
2	The new Y direction is chosen to lie in the Y-Z plane of the old coordinate system. In this setting the angle difference between the old and new X axis will be minimal.
3	The value chosen is the mean value of the two settings, which would have been chosen with 1 and 2.
A detailed description of all parameterization options can be found in: **References** Detailed Machine Data Description	

Special features and extensions

The behavior for value = 1 and 2 is achieved when starting from an arbitrary position of the X and Y axes, by rotating the coordinate system around the Z axis until the desired setting is reached.

The behavior for value = 3 is achieved by executing a rotation whose value is the exact mean of these two angles. However, this only applies for the case that the old and new Z direction enclose an angle of less than 90 degrees.

With value = 1, both the old and new X axes form an angle of under 90 degrees, with value = 2 the same is true of the Y axis (the relevant axes point in "approximately" the same direction). If the two Z directions form an angle of more than 90 degrees, however, the conditions of an angle < 90 degrees between the old and new axes can no longer be met simultaneously for both X and Y. In this case, priority is given to the X direction, i.e. a mean value is taken from the direction for 1 and the negative direction for 2.

If one of the commands TOFRAMEX, TOFRAMEY, TOROTX, TOROTY is programmed in place of TOFRAME(Z) or TOROT(Z), the descriptions for adapting the axis directions perpendicular to the main direction are also valid for the cyclically exchanged axes. The assignments in the table below are then valid:

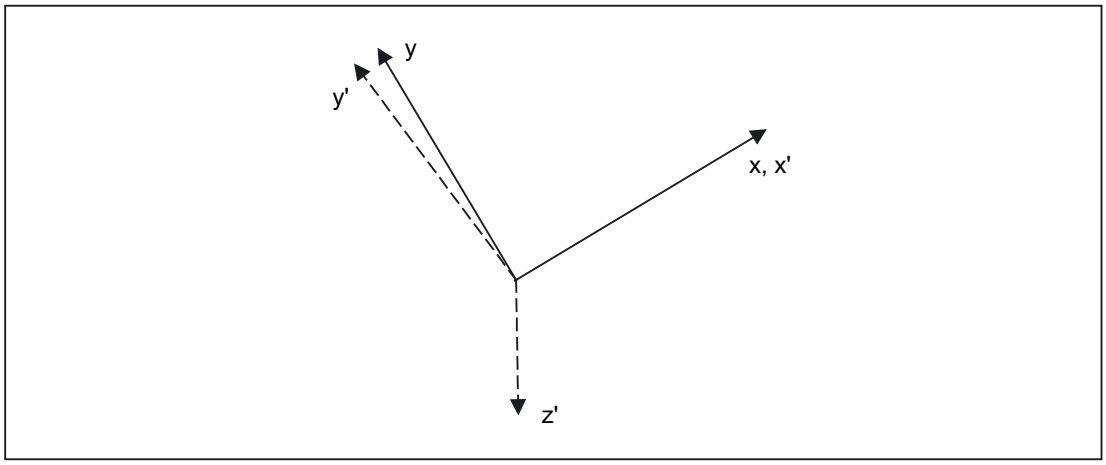
	TOFRAME, TOFRAMEZ TOROT, TOROTZ	TOFRAMEY TOROTY	TOFRAMEX TOROTX
Tool direction (vertical axis)	Z	Y	X
Secondary axis (horizontal axis)	X	Z	Y
Secondary axis (ordinate)	Y	X	Z

Example

```
N90 $SC_TOFRAME_MODE=1
N100 ROT Z45
N110 TCARR=1 TCOABS T1 D1
N120 TOROT
```

N100 describes a rotation by 45 degrees in the X-Y plane. It is assumed that the toolholder activated in N110 rotates the tool by 30 degrees around the X axis, i.e. the tool lies in the Y-Z plane and is rotated by 30 degrees relative to the Z axis. As a result the Z axis of the frame newly defined in N120 also points in this direction (independently of the value in setting data SD42980 $SC_TOFRAME_MODE in N90).

SD42980 $SC_TOFRAME_MODE == 1:



The old and new X axes X and X' coincide in the projection in the direction of the old Z axis. The old and new Y axes Y and Y' form an angle of 8.13 degrees (right angles are generally not retained in the projection).

SD42980 $SC_TOFRAME_MODE == 2:
Y and Y' coincide and X and X' form an angle of 8.13 degrees.

SD42980 $SC_TOFRAME_MODE == 3:
X and X' and well as Y and Y' each form an angle of 4.11 degrees.

Note

The named angles (8.13 and 4.11 degrees) are the angles, which the projections of the axes form in the X-Y plane. They are not the spatial angles of these axes.

TCARR (request toolholder) and PAROT (align workpiece coordinate system on the workpiece)

TCARR uses the basic frame identified by following machine data:
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER.

A system frame can be created for TCARR and PAROT alone, in order to avoid conflicts with systems, which already use all the basic frames.

PAROT, TOROT and TOFRAME have previously changed the rotation component of the programmable frames. In this case, it is not possible to shut down PAROT or TOROT separately. On RESET, the programmable frame is deleted, which means that after changing the operating mode to JOG, the rotation component of PAROT and TOROT is no longer available. The user must also have unrestricted access to the programmable frame. Frames produced by PAROT and TOROT must be able to be archived and reloaded via data backup.

The system frame for TCARR and PAROT is configured with:
MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 2 = 1

The following machine data is then no longer evaluated:
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER

If the system frame for TCARR is configured, TCARR and PAROT describe that corresponding system frame; otherwise the basic frame identified by machine data MD20184 is described.

With kinematics systems of the types P and M, TCARR will enter the table offset of the orientational toolholder (zero offset resulting from the rotation of the table) as a translation into the system frame. PAROT converts the system frame so that a workpiece-related WCS results.

The system frames are stored retentively and therefore retained after a reset. The system frames also remain active in the case of a mode change.

For the display, the commands PAROT and TOROT, TOFRAME are each assigned to a separate G code group.

PAROTOF

PAROTOF is the switch off command for PAROT. This command deletes the rotations in the system frame for PAROT. In so doing, the rotations in the current $P_PARTFRAME and in the data management frame $P_PARTFR are deleted. The position of the coordinate system is then recreated according to TCARR. PAROTOF is in the same G group as PAROT and appears therefore in the G code display.

TOROT (align Z axis of the WCS by rotating the frame parallel to the tool orientation) and TOFRAME (ditto.)

The system frame for TOROT and TOFRAME is activated via the following machine data:
MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 3 = 1

This system frame is located before the programmable frame in the frame chain.
The SZS coordinate system is located before the programmable frame.

TOROTOF

TOROTOF is the switch off command for TOROT and TOFRAME. This command deletes the corresponding system frame. The current $P_TOOLFRAME and the data management frame $P_TOOLFR are also deleted. TOROTOF is in the same G group as TOROT and TOFRAME and appears therefore in the G code display.

Example

Example of using an orientational toolholder with deactivated kinematics:

```
N10          $TC_DP1[1,1]= 120
N20          $TC_DP3[1,1]= 13                ; Tool length 13 mm
; Definition of toolholder 1:
N30          $TC_CARR1[1] = 0                ; X component of the 1st
                                             ; offset vector
N40          $TC_CARR2[1] = 0                ; Y component of the 1st
                                             ; offset vector
N50          $TC_CARR3[1] = 0                ; Z component of the 1st
                                             ; offset vector
N60          $TC_CARR4[1] = 0                ; X component of the 2nd
                                             ; offset vector
N70          $TC_CARR5[1] = 0.               ; Y component of the 2nd
                                             ; offset vector
N80          $TC_CARR6[1] = -15.             ; Z component of the 2nd
                                             ; offset vector
N90          $TC_CARR7[1] = 1                ; X components of 1st axis
N100         $TC_CARR8[1] = 0                ; Y components of 1st axis
N110         $TC_CARR9[1] = 0                ; Z components of 1st axis
N120         $TC_CARR10[1] = 0               ; X components of 2nd axis
N130         $TC_CARR11[1] = 1               ; Y components of 2nd axis
N140         $TC_CARR12[1] = 0               ; Z components of 2nd axis
N150         $TC_CARR13[1] = 30.             ; Angle of rotation of 1st axis
N160         $TC_CARR14[1] = -30.            ; Angle of rotation of 2nd axis
N170         $TC_CARR15[1] = 0               ; X component of the 3rd
                                             ; offset vector
N180         $TC_CARR16[1] = 0               ; Y component of the 3rd
                                             ; offset vector
N190         $TC_CARR17[1] = 0               ; Z component of the 3rd
                                             ; offset vector
N200         $TC_CARR18[1] = 0               ; X component of the 4th
                                             ; offset vector
N210         $TC_CARR19[1] = 0               ; Y component of the 4th
                                             ; offset vector
N220         $TC_CARR20[1] = 15.             ; Z component of the 4th
                                             ; offset vector
N230         $TC_CARR21[1] = A               ; Reference for 1st axis
N240         $TC_CARR22[1] = B               ; Reference for 2nd axis
N250         $TC_CARR23[1] = "M"             ; Toolholder type
N260         X0 Y0 Z0 A0 B45 F2000
N270         TCARR=1 X0 Y10 Z0 T1 TCOABS     ; Selection of orientable
                                             ; toolholder
```

```
N280        PAROT                          ; Rotation of table
N290        TOROT                          ; Rotation of the z axis in
                                           ; tool orientation
N290        X0 Y0 Z0
N300        G18 MOVT=AC(20)                ; Processing in G18 plane
N310        G17 X10 Y0 Z0                  ; Processing in G17 plane
N320        MOVT=-10
N330        PAROTOF                        ; Deactivate rotation of table
N340        TOROTOF                        ; No longer align WCS to
                                           ; tool.
N400        M30
```

9.5.9 Subprograms with SAVE attribute (SAVE)

For various frames, the behavior regarding subprograms can be set using the SAVE attribute.

Settable frames G54 to G599

The behavior of the adjustable frames can be set using MD10617 $MN_FRAME_SAVE_MASK.**BIT0** :

- BIT0 = 0

  Using the subprogram, if only the values of the active adjustable frame are changed using the system variable $P_IFRAME, but the G functions are kept, then the change is also kept after the end of the subprogram.

- BIT0 = 1

  With the end of the subprogram, the adjustable frame, G function and values, active before the subprogram call, are reactivated.

Basic frames $P_CHBFR[ ] and $P_NCBFR[ ]

The behavior of the basic frame can be set using MD10617 $MN_FRAME_SAVE_MASK.**BIT1**:

- BIT1 = 0
  If the active basic frame is changed by the subprogram, the change remains effective even after the end of the subprogram.

- BIT1 = 1
  With the end of the subprogram the basic frame which is active before the subroutine call is reactivated.

Programmable frame

With the end of the subprogram the programmable frame active before the subroutine call is reactivated.

System frames

If the system frames are changed by the subprogram, the change remains effective even after the end of the subprogram.

9.5.10 Data backup

Data block _N_CHANx_UFR is used to archive the system frames.

Machine data
MD28082 $MC_MM_SYSTEM_FRAME_MASK
should not have changed between saving and reintroducing the saved system frames. If it has changed then it is possible that saved system frames could no longer be loaded.
In this case, the loading process triggers an alarm.

Data backup always takes place in accordance with the currently valid geometry axis assignment, not in accordance with the axis configurations set in the machine data.

The machine data
$MC_MM_SYSTEM_DATAFRAME_MASK
can be used to configure data management frames for the system frames.
If you do not want a data management frame for a system frame, the frame does not have to be saved. With `G500`, `G54` to `G599`, the active frame is retained.

A separate data block _N_NC_UFR is used to archive global frames.
The block requested by the HMI is created if the machine data
MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES
or
MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES
has a value greater than zero.
Channel-specific frames are saved in data block _N_CHANx_UFR.

In certain circumstances, alarms could be triggered when reintroducing saved data, if the frame affiliates, be they NCU global or channel-specific, have been changed using machine data.

Data backup always takes place in accordance with the axis configuration set in the machine data, not in accordance with the currently valid geometry axis assignment.

9.5.11 Positions in the coordinate system

The setpoint positions in the coordinate system can be read via the following system variables. The actual values can be displayed in the WCS, SZS, BZS or MCS via the PLC. There is a softkey for actual-value display in MCS/WCS. Machine manufacturers can define on the PLC side, which coordinate system corresponds to the workpiece coordinate system on their machines. The HMI requests the appropriate actual values from the NCK.

$AA_IM[axis]

The setpoints in the machine coordinate system can be read for each axis using the variables $AA_IM[axis].

$AA_IEN[axis]

The setpoints in the settable zero system (SZS) can be read for each axis using the variables $AA_IEN[axis].

$AA_IBN[axis]

The setpoints in the basic zero system (BZS) can be read using the variable $AA_IBN[axis].

$AA_IW[axis]

The setpoints in the workpiece coordinate system (WCS) can be read using the variable $AA_IW[axis].

9.5.12 Control system response

9.5.12.1 POWER ON

Frame conditions after POWER ON

Frame	Frame conditions after POWER ON
Programmable frame	Deleted.
Settable frames	Are retained, depending on: MD24080 $MC_USER_FRAME_POWERON_MASK Bit 0 = 1 MD20152 $MC_GCODE_RESET_MODE[7] = 1
Complete basic frame	Retained, depending on MD20110 $MC_RESET_MODE_MASK bit 0 and bit 14 Individual basic frames can be deleted with MD10615 $MN_NCBFRAME_POWERON_MASK and MD24004 $MC_CHBFRAME_POWERON_MASK .
System frames	Retained Depending on: MD24008 $MC_CHSFRAME_POWERON_MASK , individual system frames can be deleted on POWER ON. Deletion of system frame is executed in the data management on first priority.
Zero offset external	Permanent, but has to be activated again. The system frame is retained.
DRF offset	Deleted.

9.5.12.2 Mode change

System frames

System frames are retained and remain active when the operating mode is changed.

JOG mode

In JOG mode, the frame components of the current frame are only taken into account for the geometry axes if a rotation is active. No other axial frames are taken into account.

PLC and command axes

The response for PLC and command axes can be set via machine data:

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED (Frame or HL correction is not permissible)

9.5.12.3 RESET, end of part program

RESET responses of basic frames

The RESET response of basic frames is set via the machine data:

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

RESET responses of system frames

The system frames are retained in the data management after a Reset.

The machine data below can be used to configure the activation of individual system frames:

MD24006 $MC_CHSFRAME_RESET_MASK (active system frames after Reset)

Bit	Significance
0	System frame for actual value setting and scratching is active after RESET.
1	System frame for zero offset external is active after RESET.
2	Is not evaluated.
3	Is not evaluated.
4	System frame for workpiece reference point is active after RESET.
5	System frame for cycles is active after RESET.
6	Reserved, RESET response depends on MD20110 $MC_RESET_MODE_MASK.
7	System frame $P_ISO1FR is active after RESET.
8	System frame $P_ISO2FR is active after RESET.
9	System frame $P_ISO3FR is active after RESET.
10	System frame $P_ISO4FR is active after RESET.
11	System frame $P_RELFR is active after RESET.

RESET response of the system frames of TCARR, PAROT, TOROT and TOFRAME

The RESET response of the system frames of `TCARR`, `PAROT`, `TOROT` and `TOFRAME` depends on the G-Code RESET setting.

The setting is made with machine data:

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

MD20152 $MC_GCODE_RESET_MODE[ ] (RESET response of G groups)

MD20150 $MC_GCODE_RESET_VALUES (RESET position of G groups)

MD20110	Significance	
Bit 0 = 0	TCARR and PAROT system frames are retained as before the RESET.	
Bit 0 = 1	MD20152 $MC_GCODE_RESET_MODE[51] = 0	
	MD20150 $MC_GCODE_RESET_VALUES[51] = 1	PAROTOF
	MD20150 $MC_GCODE_RESET_VALUES[51] = 2	PAROT
	MD20152 $MC_GCODE_RESET_MODE[51] = 1	TCARR and PAROT system frames are retained as before the RESET.
	MD20152 $MC_GCODE_RESET_MODE[52] = 0	
	MD20150 $MC_GCODE_RESET_VALUES[52] = 1	TOROTOF
	MD20150 $MC_GCODE_RESET_VALUES[52] = 2	TOROT
	MD20150 $MC_GCODE_RESET_VALUES[52] = 3	TOFRAME
	MD20152 $MC_GCODE_RESET_MODE[52] = 1	TOROT and TOFRAME system frames are retained as before the RESET.

TCARR and PAROT are two independent functions, which describe the same frame. With PAROTOF, the component of TCARR is not activated on RESET.

MD20110	Significance	
Bit 0 = 1 and bit 14 = 0	Chained complete basic frame is deleted.	
Bit 0 = 1 and bit 14 = 1	The complete basic frame is derived on the basis of: MD24002 $MC_CHBFRAME_RESET_MASK (active channel-specific basic frame after RESET) and MD10613 $MN_NCBFRAME_RESET_MASK (active NCU global basic frame after RESET)	
	MD24002 $MC_CHBFRAME_RESET_MASK	
	Bit 0 = 1	1. Channel basic frame is calculated into the chained complete basic frame.
	Bit 7 = 1	8. Channel basic frame is calculated into the chained complete basic frame.
	MD10613 $MN_NCBFRAME_RESET_MASK	
	Bit 0 = 1	1. NCU global basic frame is calculated into the chained complete basic frame.
	Bit 7 = 1	8. NCU global basic frame is calculated into the chained complete basic frame.

Frame conditions after RESET / parts program end

Frame	condition after RESET / part program end
Programmable frame	Deleted.
Settable frames	Retained, depending on MD20110 $MC_RESET_MODE_MASK and MD20152 $MC_GCODE_RESET_MODE.
Complete basic frame	Retained, depending on: MD20110 $MC_RESET_MODE_MASK Bit 0 and Bit 14, MD10613 $MN_NCBFRAME_RESET_MASK and MD24002 $MC_CHBFRAME_RESET_MASK.
System frames	Retained, depending on MD24006 $MC_CHSFRAME_RESET_MASK and MD20150 $MC_GCODE_RESET_VALUES[ ].
External zero offset	Retained
DRF offset	Retained

Deletion of system frames

The system frames in the data management can be deleted during RESET uisng machine data:

MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK (deletion of system frames during RESET)

Bit	Significance
0	System frame for actual value setting and scratching is deleted during RESET.
1	System frame for zero offset external is deleted during RESET.
2	Reserved, for TCARR and PAROT see MD20150 $MC_GCODE_RESET_VALUES[ ].
3	Reserved, for TOROT and TOFRAME see MD20150 $MC_GCODE_RESET_VALUES[ ].
4	System frame for workpiece reference points is deleted during RESET.
5	System frame for cycles is deleted during RESET.
6	Reserved, RESET response depends on MD20110 $MC_RESET_MODE_MASK.
7	System frame for $P_ISO1FR is deleted during RESET.
8	System frame for $P_ISO2FR is deleted during RESET.
9	System frame for $P_ISO3FR is deleted during RESET.
10	System frame for $P_ISO4FR is deleted during RESET.
11	System frame $P_RELFR is deleted during RESET.

9.5.12.4 Part program start

Frame conditions after part program start

Frame	Condition after parts program start
Programmable frame	Deleted.
Settable frames	Retained, depending on: MD20112 $MC_START_MODE_MASK
Complete basic frame	Retained
System frames	Retained
External zero offset	Retained
DRF offset	Retained

9.5.12.5 Block search

Block search

Data management frames are also modified when carrying out a block search with calculation.

Cancellation of block search

If the block search is aborted with RESET, then the machine data:
MD28560 $MC_MM_SEARCH_RUN_RESTORE_MODE
can be used to configure that all data management frames are set to the value they had before the block search:

Bit	Significance
0	All frames in the data management are restored.

In case of cascaded block searches, the frames are set to the status of the previous block search.

SERUPRO

The "SERUPRO" function is not supported.

9.5.12.6 REPOS

There is no special treatment for frames. If a frame is modified in an ASUB, it is retained in the program. On repositioning with REPOS, a modified frame is included, provided the modification was activated in the ASUB.

9.6 Workpiece-related actual value system

9.6.1 Overview

Definition

The term "workpiece-related actual-value system" designates a series of functions that permit the user:

- To use a workpiece coordinate system defined in machine data after powerup.

  Features:

  – No additional operations are necessary.

  – Effective in JOG and AUTOMATIC modes

- To retain the valid settings for the following after end of program for the next part program:

  – Active plane

  – Settable frame (G54-G57)

  – Kinematic transformation

  – Active tool offset

- To change between work coordinate system and machine coordinate system via the user interface.
- To change the work coordinate system by operator action (e.g., changing the settable frame or the tool offset).

9.6.2 Use of the workpiece-related actual value system

Requirements, basic settings

The settings described in the previous Section have been made for the system.
The predefined setting after power-up of the HMI software is MCS.

Switchover to WCS

The change to the WCS via the user interface causes the axis positions relative to the origin of the WCS to be displayed.

Switchover to MCS

The change to the MCS via the user interface causes the axis positions relative to the origin of the MCS to be displayed.

Interrelationships between coordinate systems

The figure below shows the interrelationships between the machine coordinate system (MCS) and the workpiece coordinate system (WCS).

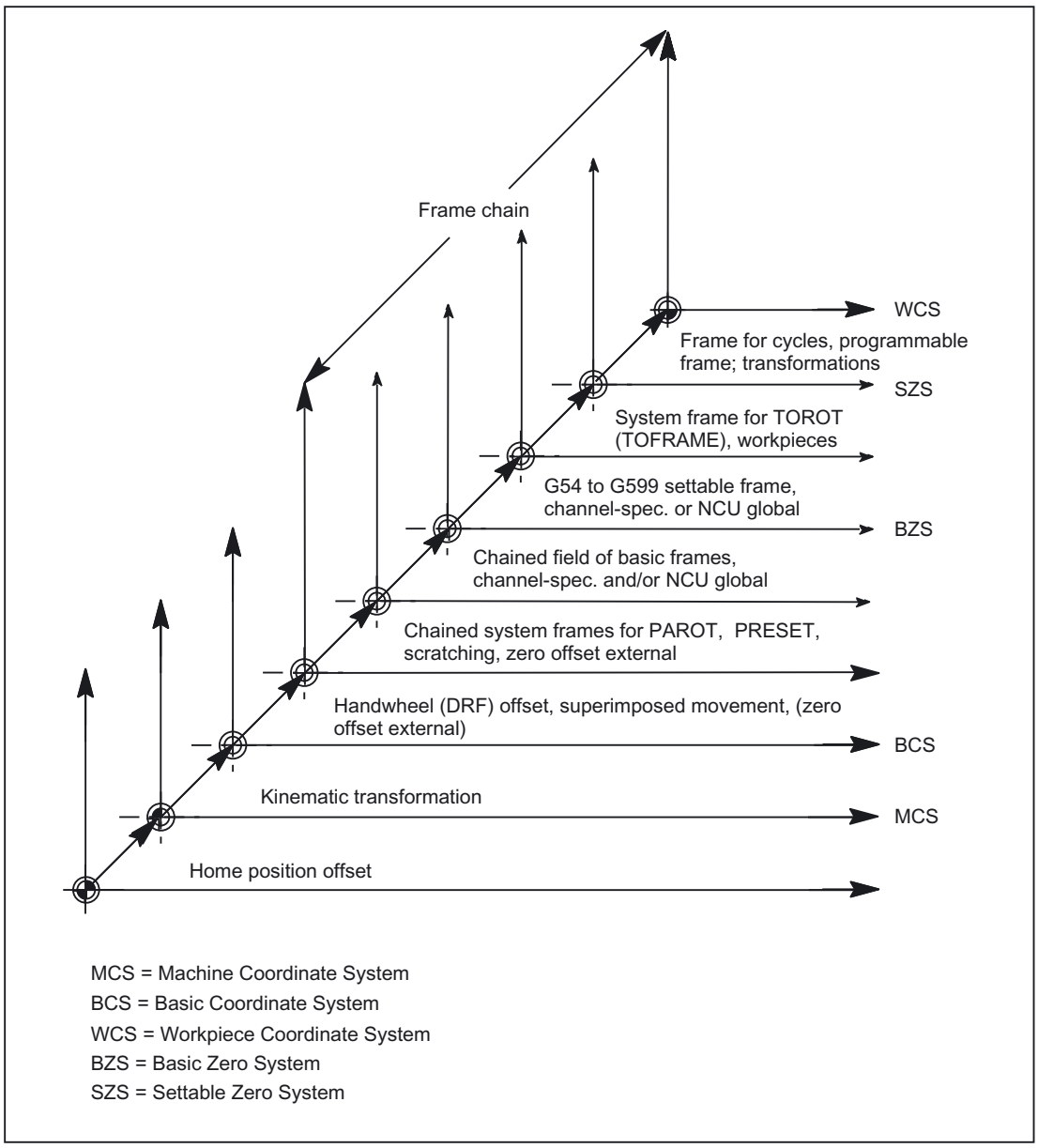


Figure 9-26 Interrelationship between coordinate systems

For further information, see "H2: Auxiliary function outputs to PLC (Page 395)" and "W1: Tool offset (Page 1567)".

References:

- Programming Guide Fundamentals
- Function Manual, Extended Functions; Kinematic Transformation (M1)
- Function Manual, Special Functions; Axis Couplings and ESR (M3);
  Section: Coupled motion, Section: Master value coupling
- Function Manual, Special Functions; Tangential Control (T3)

9.6.3 Special reactions

Overstore

Overstoring in RESET state of:

- Frames (zero offsets)
- Active plane
- Activated transformation
- Tool offset

immediately affects the actual-value display of all axes in the channel.

Entry via operator panel front

If operations on the operator panel are used to change the values for
"Active frame" (zero offsets, "Parameters" operating area)
and
"Active tool length compensation" ("Parameters" operating area)
, one of the following actions is used to activate these changes in the display:

- Press the RESET key.
- Reselect:
  - Zero offset by the part program
  - Tool offset by the part program
- Reset:
  - Zero offset by overstoring
  - Tool offset by overstoring
- Part program start

MD9440

If the HMI machine data
MD9440 ACTIVATE_SEL_USER_DATA
for the operator panel front is set, the entered values become active immediately in RESET state.

When values are entered in the part-program execution stop state, they become effective when program execution continues.

Actual-value reading

If the actual value of $AA_IW is read in the WCS after activation of a frame (zero offset) or a tool offset, the activated changes are already contained in the result read even if the axes have not yet been traversed with the activated changes.

The actual values in the settable zero system (SZS) can be read from the part program for each axis using the variable $AA_IEN[axis].

The actual values in the basic zero system (BZS) can be read from the part program for each axis using the variable $AA_IBN[axis].

Actual-value display

The programmed contour is always displayed in the WCS.

The following offsets are added to the MCS:

- Kinematic transformation
- DRF offset/zero offset external
- Active frame
- Active tool offset of the current tool

Switchover by PLC

The actual values can be displayed in the WCS, SZS, BZS or MCS via the PLC. The PLC can define, which coordinate system corresponds to the workpiece coordinate system on a machine.

On MMC power-up the MCS is preset.
With the signal DB19 DBB0.7 "MCS/WCS switchover", it is also possible to switch from the PLC to the WCS.

Transfer to PLC

Depending on machine data
MD20110 / MD20112, bit 1
, the auxiliary functions (D, T, M) are output to the PLC (or not) on selection of the tool length compensation.

Note

If the WCS is selected from the PLC, an operator action can still be used to switch between the WCS and MCS for the relevant mode.
However, when the mode and or area is changed, the WCS selected by the PLC is evaluated and activated (see Section "K1: Mode group, channel, program operation, reset response (Page 489)").

9.7 Restrictions

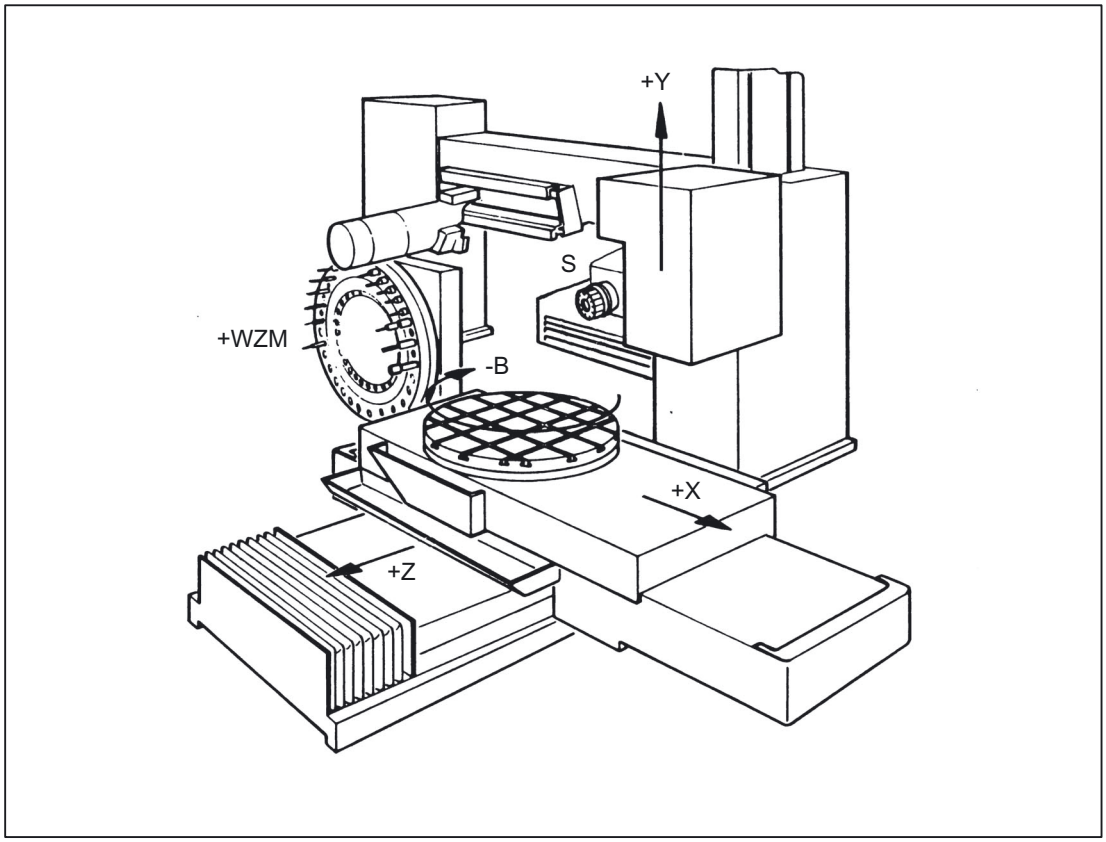
There are no supplementary conditions to note.

9.8 Examples

9.8.1 Axes

Axis configuration for a 3axis milling machine with rotary table

1. Machine axis: X1	Linear axis
2. Machine axis: Y1	Linear axis
3. Machine axis: Z1	Linear axis
4. Machine axis: B1	Rotary table (for turning for multiface machining)
5. Machine axis: W1	Rotary axis for tool magazine (tool revolver)
6. Machine axis: C1	(Spindle)
1. Geometry axis: X	(1. channel)
2. Geometry axis: Y	(1. channel)
3. Geometry axis: Z	(1. channel)
1. Special axis: B	(1. channel)
2. Special axis: WZM	(1. channel)
1. spindle: S1/C	(1. channel)



Parameterization of the machine data

Machine data	Value
MD10000 AXCONF_MACHAX_NAME_TAB[0]	= X1
MD10000 AXCONF_MACHAX_NAME_TAB[1]	= Y1
MD10000 AXCONF_MACHAX_NAME_TAB[2]	= Z1
MD10000 AXCONF_MACHAX_NAME_TAB[3]	= B1
MD10000 AXCONF_MACHAX_NAME_TAB[4]	= W1
MD10000 AXCONF_MACHAX_NAME_TAB[5]	= C1
MD20050 AXCONF_GEOAX_ASSIGN_TAB[0]	= 1
MD20050 AXCONF_GEOAX_ASSIGN_TAB[1]	= 2
MD20050 AXCONF_GEOAX_ASSIGN_TAB[2]	= 3
MD20060 AXCONF_GEOAX_NAME_TAB[0]	=X
MD20060 AXCONF_GEOAX_NAME_TAB[1]	=Y
MD20060 AXCONF_GEOAX_NAME_TAB[2]	=Z
MD20070 AXCONF_MACHAX_USED[0]	= 1
MD20070 AXCONF_MACHAX_USED[1]	= 2

Machine data	Value
MD20070 AXCONF_MACHAX_USED[2]	= 3
MD20070 AXCONF_MACHAX_USED[3]	= 4
MD20070 AXCONF_MACHAX_USED[4]	= 5
MD20070 AXCONF_MACHAX_USED[5]	= 6
MD20080 AXCONF_CHANAX_NAME_TAB[0]	=X
MD20080 AXCONF_CHANAX_NAME_TAB[1]	=Y
MD20080 AXCONF_CHANAX_NAME_TAB[2]	=Z
MD20080 AXCONF_CHANAX_NAME_TAB[3]	= B
MD20080 AXCONF_CHANAX_NAME_TAB[4]	= WZM
MD20080 AXCONF_CHANAX_NAME_TAB[5]	= S1
MD30300 IS_ROT_AX[3]	= 1
MD30300 IS_ROT_AX[4]	= 1
MD30300 IS_ROT_AX[5]	= 1
MD30310 ROT_IS_MODULO[3]	= 1
MD30310 ROT_IS_MODULO[4]	= 1
MD30310 ROT_IS_MODULO[5]	= 1
MD30320 DISPLAY_IS_MODULO[3]	= 1
MD30320 DISPLAY_IS_MODULO[4]	= 1
MD20090 SPIND_DEF_MASTER_SPIND	= 1
MD35000 SPIND_ASSIGN_TO_MACHAX[AX1]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX2]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX3]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX4]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX5]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX6]	= 1

9.8.2 Coordinate systems

Configuring a global basic frame

An NC with 2 channels is required. The following applies:

- The global basic frame can then be written by either channel.
- The other channel recognizes this change when the global basic frame is reactivated.
- The global basic frame can be read by either channel.
- Either channel can activate the global basic frame for that channel.

Machine data

Machine data	Value
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]	= X1
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]	= X2
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]	= X3
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]	= X4
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4]	= X5
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[5]	= X6
MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES	= 1
MD28081 $MC_MM_NUM_BASE_FRAMES	= 1

Machine data for channel 1	Value		Machine data for channel 1	Value
$MC_AXCONF_CHANAX_NAME_TAB[0]	=X		$MC_AXCONF_CHANAX_NAME_TAB[0]	=X
$MC_AXCONF_CHANAX_NAME_TAB[1]	=Y		$MC_AXCONF_CHANAX_NAME_TAB[1]	=Y
$MC_AXCONF_CHANAX_NAME_TAB[2]	=Z		$MC_AXCONF_CHANAX_NAME_TAB[2]	=Z
$MC_AXCONF_MACHAX_USED[0]	= 1		$MC_AXCONF_MACHAX_USED[0]	= 4
$MC_AXCONF_MACHAX_USED[1]	= 2		$MC_AXCONF_MACHAX_USED[1]	= 5
$MC_AXCONF_MACHAX_USED[2]	= 3		$MC_AXCONF_MACHAX_USED[2]	= 6
$MC_AXCONF_GEOAX_NAME_TAB[0]	=X		$MC_AXCONF_GEOAX_NAME_TAB[0]	=X
$MC_AXCONF_GEOAX_NAME_TAB[1]	=Y		$MC_AXCONF_GEOAX_NAME_TAB[1]	=Y
$MC_AXCONF_GEOAX_NAME_TAB[2]	=Z		$MC_AXCONF_GEOAX_NAME_TAB[2]	=Z
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]	= 1		$MC_AXCONF_GEOAX_ASSIGN_TAB[0]	= 4
$MC_AXCONF_GEOAX_ASSIGN_TAB[1]	= 2		$MC_AXCONF_GEOAX_ASSIGN_TAB[1]	= 5
$MC_AXCONF_GEOAX_ASSIGN_TAB[2]	= 3		$MC_AXCONF_GEOAX_ASSIGN_TAB[2]	= 6

Part program in first channel

```
Code (excerpt)                                  Comment
. . .
N100 $P_NCBFR[0] = CTRANS( x, 10 )        ;   Activation of the NC global basic frame
. . .
N130 $P_NCBFRAME[0] = CROT(X, 45)         ;   Activation of the NC global basic frame with rotation =>
                                              alarm 18310, since rotations of NC global frames
                                              are not permitted
. . .
```

Part program in second channel

```
Code (excerpt)                                  Comment
. . .
N100 $P_NCBFR[0] = CTRANS( x, 10 )        ;   The NCU global basic frame is also active in second
                                              channel.
. . .
N510 G500 X10                             ;   Activate basic frame
N520 $P_CHBFRAME[0] = CTRANS( x, 10 )     ;   Current frame of second channel is activated with an
                                              offset.
. . .
```

9.8.3 Frames

Example 1

The channel axis is to become a geometry axis through geometry axis substitution.

The substitution is to give the programmable frame a translation component of 10 in the X axis.

The current settable frame is to be retained:

FRAME_GEOX_CHANGE_MODE = 1

```
$P_UIFR[1] =                              ; Frame is retained after geo axis substitution.
CROT(x,10,y,20,z,30)
G54                                       ; Settable frame becomes active.
TRANS a10                                 ; Axial offset of a is also substituted.
GEOAX(1, a)                               ; a becomes x axis.
                                          ; $P_ACTFRAME= CROT(x,10,y,20,z,30):CTRANS(x10)
```

Several channel axes can become geometry axes on a transformation change.

Example 2

Channel axes 4, 5 and 6 become the geometry axes of a 5axis orientation transformation. The geometry axes are thus all substituted before the transformation.

The current frames are changed when the transformation is activated.

The axial frame components of the channel axes, which become geometry axes, are taken into account when calculating the new WCS. Rotations programmed before the transformation are retained. The old WCS is restored when the transformation is deactivated.

The most common application will be that the geometry axes do not change before and after the transformation and that the frames are to stay as they were before the transformation.

Machine data:

```
$MN_FRAME_GEOAX_CHANGE_MODE = 1

$MC_AXCONF_CHANAX_NAME_TAB[0] = "CAX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "CAY"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "CAZ"
$MC_AXCONF_CHANAX_NAME_TAB[3] = "A"
$MC_AXCONF_CHANAX_NAME_TAB[4] = "B"
$MC_AXCONF_CHANAX_NAME_TAB[5] = "C"

$MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
$MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2
$MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"
$MC_AXCONF_GEOAX_NAME_TAB[1]="Y"
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=5
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=6

$MC_TRAFO_AXES_IN_1[0] = 4
$MC_TRAFO_AXES_IN_1[1] = 5
$MC_TRAFO_AXES_IN_1[2] = 6
$MC_TRAFO_AXES_IN_1[3] = 1
$MC_TRAFO_AXES_IN_1[4] = 2
```

Program:

```
$P_NCBFRAME[0] = ctrans(x,1,y,2,z,3,a,4,b,5,c,6)
$P_CHBFRAME[0] = ctrans(x,1,y,2,z,3,a,4,b,5,c,6)
$P_IFRAME = ctrans(x,1,y,2,z,3,a,4,b,5,c,6):crot(z,45)
$P_PFRAME = ctrans(x,1,y,2,z,3,a,4,b,5,c,6):crot(x,10,y,20,z,30)

TRAORI                          ; Geo axis (4,5,6) sets transformer
                                ; $P_NCBFRAME[0] =
                                  ctrans(x,4,y,5,z,6,cax,1,cay,2,caz,3)
                                ; $P_ACTBFRAME =
                                  ctrans(x,8,y,10,z,12,cax,2,cay,4,caz,6)
                                ; $P_PFRAME = ctrans(x,4,y,5,z,6,cax,1,cay,2,caz,3):
                                ; crot(x,10,y,20,z,30)
                                ; $P_IFRAME =
                                  ctrans(x,4,y,5,z,6,cax,1,cay,2,caz,3):crot(z,45)

TRAFOOF;                        ; Geo axis (1,2,3) sets transformation deactivation
                                ; $P_NCBFRAME[0] = ctrans(x,1,y,2,z,3,a,4,b,5,c,6)
                                ; $P_CHBFRAME[0] = ctrans(x,1,y,2,z,3,a,4,b,5,c,6)
                                ; $P_IFRAME =
                                  ctrans(x,1,y,2,z,3,a,4,b,5,c,6):crot(z,45)
                                ; $P_PFRAME =
                                  ctrans(x,1,y,2,z,3,a,4,b,5,c,6):crot(x,10,y,20,z,30)
```

9.9 Data lists

9.9.1 Machine data

9.9.1.1 Displaying machine data

Number	Identifier: $MM_	Description
SINUMERIK Operate		
9242	MA_STAT_DISPLAY_BASE	Numerical basis for display of moving joint STAT
9243	MA_TU_DISPLAY_BASE	Numerical basis for display of rotary axis position TU
9244	MA_ORIAXES_EULER_ANGLE_NAME	Display of orientation axes as Euler angle
9245	MA_PRESET_FRAMEIDX	Value storage scratching and PRESET
9247	USER_CLASS_BASE_ZERO_OFF_PA	Availability of basic offset in "Parameters" operating area
9248	USER_CLASS_BASE_ZERO_OFF_MA	Availability of basic offset in Machine operating area
9424	MA_COORDINATE_SYSTEM	Coordinate system for actualvalue display
9440	ACTIVE_SEL_USER_DATA	Active data (frames) are immediately operative after editing
9449	WRITE_TOA_LIMIT_MASK	Applicability of MD9203 to edge data and locationdependent offsets
9450	MM_WRITE_TOA_FINE_LIMIT	Limit value for wear fine
9451	MM_WRITE_ZOA_FINE_LIMIT	Limit value for offset fine

9.9.1.2 NC-specific machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB	Machine axis name
10600	FRAME_ANGLE_INPUT_MODE	Sequence of rotation in the frame
10602	FRAME_GEOAX_CHANGE_MODE	Frames and switchover of geometry axes
10610	MIRROR_REF_AX	Reference axis for mirroring
10612	MIRROR_TOGGLE	Change over mirror
10613	NCBFRAME_RESET_MASK	ActiveNCU-global basic frame after reset
10615	NCBFRAME_POWERON_MASK	Reset global basic frames after Power On
10617	FRAME_SAVE_MASK	Behavior of frames for SAVE subprograms
10650	IPO_PARAM_NAME_TAB	Name of interpolation parameters
10660	INTERMEDIATE_POINT_NAME_TAB	Name of intermediate point coordinates for G2/G3
11640	ENABLE_CHAN_AX_GAP	Channel axis gaps are allowed
18600	MM_FRAME_FINE_TRANS	Fine offset for FRAME (SRAM)
18601	MM_NUM_GLOBAL_USER_FRAMES	Number of globally predefined user frames (SRAM)
18602	MM_NUM_GLOBAL_BASE_FRAMES	Number of global basic frames (SRAM)

9.9.1.3 Channel-specific machine data

Number	Identifier: $MC_	Description
20050	AXCONF_GEOAX_ASSIGN_TAB	Assignment geometry axis to channel axis
20060	AXCONF_GEOAX_NAME_TAB	Geometry axis name in channel
20070	AXCONF_MACHAX_USED	Machine axis number valid in channel
20080	AXCONF_CHANAX_NAME_TAB	Channel axis name in the channel
20110	RESET_MODE_MASK	Definition of basic control settings after RESET / TP end
20118	GEOAX_CHANGE_RESET	Allow automatic geometry axis change
20126	TOOL_CARRIER_RESET_VALUE	Active toolholder on RESET
20140	TRAFO_RESET_VALUE	Transformation record on power-up (RESET / TP-End)
20150	GCODE_RESET_VALUES	Initial setting of the G groups
20152	GCODE_RESET_MODE	RESET response of the G groups
20184	TOCARR_BASE_FRAME_NUMBER	Number of the basic frame for pickup of the table offset
21015	INVOLUTE_RADIUS_DELTA	End point monitoring for evolvents (involutes)
22532	GEOAX_CHANGE_M_CODE	M code for replacement of geometry axes
22534	TRAFO_CHANGE_M_CODE	M code for transformation change
24000	FRAME_ADD_COMPONENTS	Frame components for G58 and G59
24002	CHBFRAME_RESET_MASK	RESET response of channel-specific basic frames

Number	Identifier: $MC_	Description
24004	CHBFRAME_POWERON_MASK	Reset channel-specific basic frames after Power On
24006	CHSFRAME_RESET_MASK	Active system frames after reset
24007	CHSFRAME_RESET_CLEAR_MASK	Clear system frames on RESET
24008	CHSFRAME_POWERON_MASK	Reset system frames after POWER ON
24010	PFRAME_RESET_MODE	RESET mode for programmable frame
24020	FRAME_SUPPRESS_MODE	Positions for frame suppression
24030	FRAME_ACT_SET	SZS coordinate system setting
24040	FRAME_ADAPT_MODE	Adapting active frames
24050	FRAME_SAA_MODE	Saving and activating data management frames
24805	TRACYL_ROT_AX_FRAME_1	Rotary axis offset TRACYL 1
24855	TRACYL_ROT_AX_FRAME_2	Rotary axis offset TRACYL 2
24905	TRANSMIT_ROT_AX_FRAME_1	Rotary axis offset TRANSMIT1
24955	TRANSMIT_ROT_AX_FRAME_2	Rotary axis offset TRANSMIT2
28080	MM_NUM_USER_FRAMES	Number of settable Frames (SRAM)
28081	MM_NUM_BASE_FRAMES	Number of basic frames
28082	MM_SYSTEM_FRAME_FRAMES	System frames (SRAM)
28560	MM_SEARCH_RUN_RESTORE_MODE	Restore data after a simulation

9.9.1.4 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32074	FRAME_OR_CORRPOS_NOTALLOWED	FRAME or HL offset is not permitted
35000	SPIND_ASSIGN_TO_MACHAX	Assignment spindle to machine axis

9.9.2 Setting data

9.9.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42440	FRAME_OFFSET_INCR_PROG	Work offsets in frames
42980	TOFRAME_MODE	Frame definition for TOFRAME, TOROT and PAROT

9.9.3 System variables

Identifier	Description
$AA_ETRANS[axis]	External zero offset
$AA_IBN[axis]	Actual value in basic zero coordinate system (BZS)
$AA_IEN[axis]	Actual value in settable zero point coordinate system (SZS)
$AA_OFF[axis]	Overlaid motion for programmed axis
$AC_DRF[axis]	Handwheel override of an axis
$AC_JOG_COORD	Coordinate system for manual traversing
$P_ACSFRAME	Active frame between BCS and SZS
$P_ACTBFRAME	Active complete basic frame
$P_ACTFRAME	Active complete frame
$P_BFRAME	1st active basic frame in the channel. Corresponds to $P_CHBFRAME
$P_CHBFR[n]	Active basic frame in the channel, can be activated via G500, G54...G599
$P_CHBFRAME[n]	Active basic frame in the channel 0 to 15 NCU basic frames can be set through: MD28081 MM_NUM_BASE_FRAMES
$P_CHBFRMASK	Basic frame mask in the channel
$P_CHSFRMASK	System frame mask
$P_CYCFR	Active system frame for cycles
$P_CYCFRAME	Active system frame for cycles
$P_EXTFR	System frame for external zero offset in data management
$P_EXTFRAME	Active system frame for external zero offset
$P_IFRAME	Active settable frame
$P_ISO1FR	Data management frame for ISO G51.1 Mirroring
$P_ISO2FR	Data management frame for ISO G68 2DROT
$P_ISO3FR	Data management frame for ISO G68 3DROT
$P_ISO4FR	System frame for ISO G51 Scale
$P_ISO1FRAME	Active system frame for ISO G51.1 Mirroring
$P_ISO2FRAME	Active system frame for ISO G68 2DROT
$P_ISO3FRAME	Active system frame for ISO G68 3DROT
$P_ISO4FRAME	Active system frame for ISO G51 Scale
$P_NCBFR[n]	Global basic frame of the data management, can be activated via G500, G54...G599
$P_NCBFRAME[n]	Current NCU basic frame 0 to 15 NCU basic frames can be set through: MD18602 MM_NUM_GLOBAL_BASE_FRAMES
$P_NCBFRMASK	Global basic frame mask
$P_PARTFR	Data management frame for TCARR and PAROT
$P_PARTFRAME	Active system frame for TCARR and PAROT with orientational toolholder
$P_PFRAME	Programmable frame
$P_SETFR	Data management frame for actual value setting

Identifier	Description
$P_SETFRAME	Active system frame for actual value setting
$P_TOOLFR	Data management frame for TOROT and TOFRAME
$P_TOOLFRAME	Active system frame for TOROT and TOFRAME
$P_TRAFRAME	Data management frame for transformations
$P_TRAFRAME	Active system frame for transformations
$P_UBFR	1st basic frame in the channel in the data management that is activated after G500, G54...G599. Corresponds to $P_CHBFR[0].
$P_UIFR[n]	Settable data management frames, can be activated via G500, G54...G599
$P_UIFRNUM	Number of active settable frame $P_UIFR
$P_WPFR	Data management frame for the workpiece
$P_WPFRAME	Active system frame for the workpiece

$AA_ETRANS[X]

$AA_ETRANS[X] is an axis-specific system variable of the DOUBLE type. The default setting in the system for this variable is zero.

Values set by the user are activated through the NC/PLC interface signal:

DB31, ... DBX3.0 (external zero offset)

9.9.4 Signals

9.9.4.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
T function modification	DB21, ...DBX61.0-.2	-
D function modification	DB21, ...DBX62.0-.2	-
T function 1	DB21, ...DBB118-119	DB2500.DBD2000
D function 2	DB21, ...DBB129	DB2500.DBD5000
Number of active function G group 1 8 (bit int)	DB21, ...DBB208	DB3500.DBB0
Number of active function G group 2 8 (bit int)	DB21, ...DBB209	DB3500.DBB1
...	...	…
Number of active function G group 29 8 (bit int)	DB21, ...DBB236	DB3500.DBB28

9.9.4.2 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Accept zero offset external	DB31, ...DBX3.0	-

9.9.4.3 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Spindle/no axis	DB31, ...DBX60.0	DB390x.DBX0.0

N2: Emergency stop

10

10.1 Brief Description

Function

The control system supports the machine manufacturer in implementing an emergency stop function on the basis of the following functions:

- An emergency stop button is installed in a location easily accessible to the machine operator on all SINUMERIK machine control panels. The functionality of the emergency stop button includes the positive opening of electrical switching contacts and a mechanical self-activating latching/locking.
- The emergency stop request to the NC is transmitted via the NC/PLC interface on the PLC.
- In response to an emergency stop command, the NC decelerates all axes and spindles as quickly as possible (with setpoint value 0), i.e. braking at the current limit of the drives.
- In the case of an emergency stop, all machine functions controlled by the PLC can be brought to a safe state that can be set by the machine manufacturer.
- Unlatching the emergency stop button does not cancel the emergency stop state nor does it initiate a restart.
- After the emergency stop state has been canceled, it is not necessary to reference the machine axes or synchronize the spindles. The actual positions of the machine axes are continuously tracked during the emergency stop sequence.

10.2 Relevant standards

Relevant standards

Compliance with the following standards is essential for the emergency stop function:

- EN ISO 12000-1
- EN ISO 12000-2
- EN 418
- EN 60204

Emergency stop

In accordance with EN 418, an emergency stop is a function that:

- Is intended to prevent or diminish developing or existing risks to operating personnel, and damage to the machine or machined materials.
- Is triggered by a single action of a person, if the normal stop function is not suitable for it.

Hazards

In the terms of EN 418, risks may arise from:

- Functional irregularities (machine malfunctions, unacceptable properties of the material to be machined, human error, etc.).
- Normal operation.

Stadard EN ISO 12000-2

In accordance with the basic safety requirement of the EC Machinery Directive regarding emergency stop, machines must be equipped with an energency stop device.

Exceptions

No emergency stop device is required on machines:

- Where an emergency stop device would not reduce the risk, either because the shutdown time would not be reduced or because the measures to be taken would not be suitable for controlling the risk.
- That are held and operated manually.

NOTICE
The machine manufacturer is expressly directed to comply with the national and international standards. The SINUMERIK controllers support the machine manufacturer in the implementation of the emergency stop function according to the specifications in the following function description. But the responsibility for the emergency stop function (its triggering, sequence and acknowledgement) rests exclusively with the machine manufacturer.

10.3 Emergency stop control elements

Emergency stop control elements

In accordance with EN 418, emergency stop control elements must be designed so that they latch mechanically on their own and are easy for the operator and others to actuate in the event of an emergency.

The following list includes some possible types of control elements:

- Mushroom pushbutton switches
- Wires/cables, cords, rods
- Puller grips
- In special cases: Foot switches without protective covers

Emergency stop button and control

Actuation of the emergency stop button or a signal derived directly from the button must be routed to the controller (PLC) as a PLC input. In the PLC user program, this PLC input must be forwarded to the NC on the interface signal:

DB10 DBX56.1 (Emergency stop)

Resetting of the emergency stop button or a signal derived directly from the button must be routed to the controller (PLC) as a PLC input. In the PLC user program, this PLC input must be forwarded to the NC on the interface signal:

DB10 DBX56.2 (Acknowledge emergency stop)

Connection conditions

For connecting the emergency stop button see:
References:
Operator Components Manual

10.4 Emergency stop sequence

After actuation of the emergency stop control element, the emergency stop device must operate in the best possible way to prevent or minimize the danger.

"In the best possible way" means that the most favorable delay rate can be selected and the correct stop category (defined in EN 60204) can be determined according to a risk assessment.

Emergency stop sequence in the NC

The predefined (in EN 418) sequence of internal functions implemented to obtain the emergency stop state is as follows in the control system:

1. Part program execution is interrupted.

   All machine axes are braked in the relevant axis-specific parameterized time:

   MD36610 $MA_AX_EMERGENCY_STOP_TIME (time of braking ramp in event of errors)

   The maximum braking ramp that can be achieved thereby, is defned by the maximum brake current of the respective drive. The maximum brake current is achieved by setting a setpoint = 0 (fast braking).

2. Reset interface signal:

   DB11 DBX6.3 (Mode group ready)

3. Set the interface signal:

   DB10 DBX106.1 (emergency stop active)

4. Alarm 3000 "Emergency stop" is displayed.

5. After the expiry of a paramaterized delay time, the servo enables of machine axes are reset.

   The setting of the delay time is programmed in machine data:

   MD36620 $MA_SERVO_DISABLE_DELAY_TIME (OFF delay of the controller enable)

   The following setting rule must be observed: MD36620 ≥ MD36610

6. All machine axes are switched in the follow-up mode within the controller.

   The machine axes are no longer in position control.

Emergency stop sequence at the machine

The emergency stop sequence on the machine is determined solely by the machine manufacturer.

Attention should be paid to the following points in connection to the sequence on the NC:

- The process in the NC is started using the interface signal:

  DB10 DBX56.1 (Emergency stop)

  After the machine axes have come to a standstill, the power supply must be interrupted, in compliance with EN 418.

  Note

  The responsibility for interrupting the power supply rests with the machine manufacturer.

- The digital and analog outputs of the PLC I/O are not influenced by the emergency stop sequence in the NC.

  If individual outputs are required to attain a particular state or voltage level in the event of an emergency stop, the machine manufacturer must implement this in the PLC user program.

- The fast digital outputs of the NCK I/O system are not influenced by the emergency stop sequence in the NC.

  If individual outputs must assume a specific state in case of emergency stop, the machine manufacturer must transmit the desired state to the NC in the PLC user program via interface signals:

  DB10 DBB4-7

  Note

  If the sequence in the NC is not to be executed as described above, then the interface signal DB10 DBX56.1 (emergency stop) must not be set until an emergency stop state defined by the machine manufacturer in the PLC user program is reached.

  As long as the interface signal is not set and no other alarm is pending, all interface signals are operative in the NC. Any emergency stop state defined by the manufacturer (including axis-, spindle- and channel-specific emergency stop states) can therefore be assumed.

10.5 Emergency stop acknowledgement

The emergency stop control element may only be reset as a result of manual manipulation of the emergency stop control element according to EN 418.

Resetting of the emergency stop control element alone must not trigger a restart command.

A machine restart must be impossible until all of the actuated emergency stop control elements have been deliberately reset by hand.

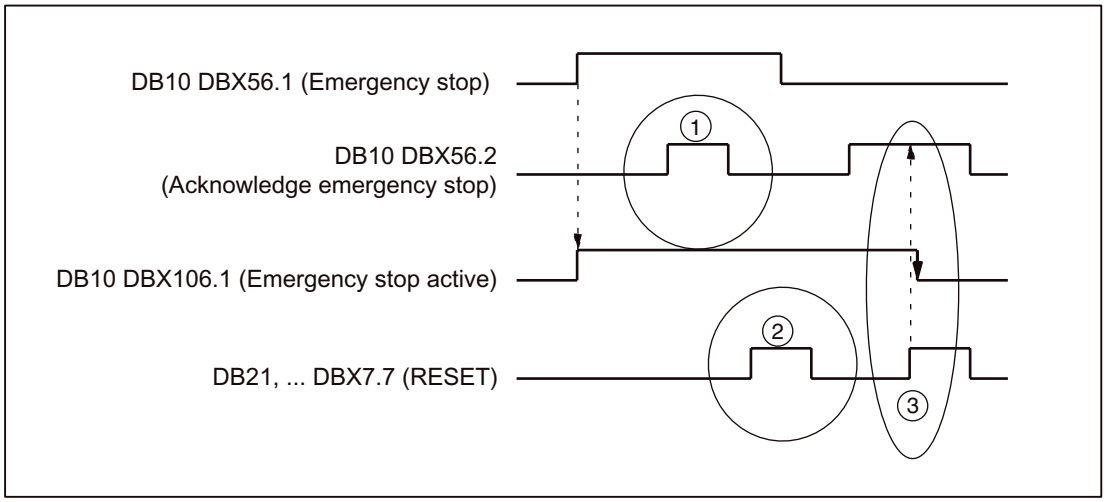
Emergency stop acknowledgement

The EMERGENCY STOP state is only reset if the interface signal:DB10 DBX56.2 (acknowledge EMERGENCY STOP) is set followed by the interface signal:DB11, ... DBX0.7 (mode group reset).

Hence it can be noted that the interface signal DB10 DBX56.2 (acknowledge emergency stop) and the interface signal DB21, ... DBX7.7 (Reset) together are set at least for so long until the interface signal DB10 DBX106.1(emergency stop active) is reset.

Note

The emergency stop state cannot be reset with the interface signal DB21, ... DBX7.7 (Reset) alone.



(1) DB10 DBX56.2 (acknowledge emergency stop) is inoperative

(2) DB21, ... DBX7.7 (Reset) is inoperative

(3) DB10 DBX56.2 and DB21, ... DBX7.7 reset DB10 DBX106.1 (emergency stop active)

Figure 10-1 Resetting the emergency stop state

Effects

Resetting the emergency stop state has the following effects:

- Within the controller for all machine axes:
  - The servo enables are set.
  - The follow-up mode is canceled.
  - The position control is activated.
- The following interface signals are set:

  DB31, ... DBX60.5 (position control active)

  DB11 DBX6.3 (mode group ready)
- The following interface signal is reset:

  DB10 DBX106.1 (emergency stop active)
- Alarm 3000 "Emergency stop" is deleted.
- Part program processing is interrupted in all channels of the NC.

PLC and NCK I/Os

The PLC user program must switch the PLC and NCK I/Os back to the state for operation of the machine.

POWER OFF / ON (supply off / on)

The emergency stop state can also be reset by switching the controller off and back on (POWER OFF / ON).

Requirement:

During power-up of the controller the interface signal DB10 DBX56.1 (emergency stop) must not be set.

10.6 Data lists

10.6.1 Machine data

10.6.1.1 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
36610	AX_EMERGENCY_STOP_TIME	Length of the braking ramp for error states
36620	SERVO_DISABLE_DELAY_TIME	Cutout delay servo enable

10.6.2 Signals

10.6.2.1 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Emergency stop	DB10.DBX56.1	DB2600.DBX0.1
Acknowledge Emergency Stop	DB10.DBX56.2	DB2600.DBX0.2

10.6.2.2 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Emergency stop active	DB10.DBX106.1	DB2700.DBX0.1

10.6.2.3 Signals to BAG

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode group RESET	DB11.DBX0.7	DB3000.DBX0.7

P1: Transverse axes

11

11.1 Brief description

Transverse axis

Within the framework of "turning" technology, the transverse axis refers to the machine axis that travels perpendicular to the axis of symmetry of the spindle, in other words, to longitudinal axis Z.

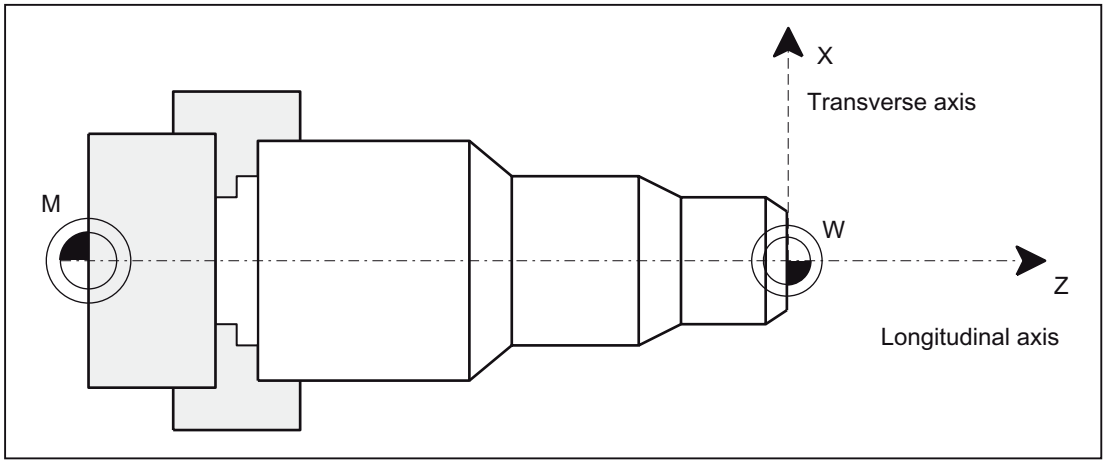


Figure 11-1 Position of the transverse axis in the machine coordinate system

Properties

- Every geometry axis of a channel can be defined as a transverse axis.
- A transverse axis is a linear axis for the following functions, which can be permitted and activated at the same time or separately:
  - Programming and display in the diameter
  - Reference axis for constant cutting speed G96/G961/G962

Several transverse axes in the channel

The introduction several transverse axes in the channel involves a functional decoupling of diameter programming and reference axis for G96/G961/G962. Diameter programming and reference axis for G96/G961/G962 can be active for different transverse axes (see table below).

	Programming and display in the diameter			**Reference axis for G96/G961/G962**	
Permissible axis type:	Geometry axis		Linear channel axes	Geometry axis	
Selection in the channel	one	m of 3	m of n	one	one of 3
Specific effect: **Machine data:**	Channel MD20100	Axis MD30460		Channel MD20100	
Programming:	DIAM* channel-specific modal G group 29			SCC[AX] channel-specific modal Reference axis for G96/G961/G962	
Acceptance during axis replacement:	DIAM*A[AX] axis-specific modal				
Axis-specific non-modal diametral/radius programming	DAC, DIC; RAC, RIC blockwise axis-specific only programming				
DIAM*: DIAMOF, DIAMON, DIAM90, DIAMCYCOF DIAM*A[AX]: DIAMOFA[AX], DIAMONA[AX], DIAM90A[AX], DIACYCOFA[AX], DIAMCHANA[AX] AX: Axis identifier for geometry/channel or machine axis identifier					

Note

Rotary axes are not permitted to serve as transverse axes.

Programming the transversing paths

The traverse paths of a transverse axis programmed in the part program may be either radius- or diameter-based. It is possible to switch between the two reference types with the part program commands DIAMON (DIAMeter ON = diameter) and DIAMOF (DIAMeter OF = radius). In this way, dimensional information can be taken directly from the technical drawing without conversion.

Active parts program

When the part program DIAMON (dimensional information as diameter) is active, the following is true for the transverse axis:

- The setpoint and actual values that refer to the workpiece coordinate system are displayed as diameter values.
- System variables for setpoints and actual values that refer to the workpiece coordinate system contain diameter values.
- Offsets are entered, programmed and displayed in radius format.
- Programmed end positions are converted to radius values internally.
- The absolute interpolation parameters (e.g. I, J, K) for circular interpolation (G2 and G3) are converted to radius values internally.
- Measurement results that were determined by touch trigger probe in the workpiece coordinate system are stored as diameter measurements.
- Setpoints and actual values can be read in diameter format in the WCS with the aid of system variables.

When the part program command DIAMOF (dimensional information as radius) is active, the above-mentioned data is always entered, programmed, internally stored, read or displayed as radius data.

11.2 Defining a geometry axis as transverse axis

Definition of a transversing axis in the channel

The definition of **one** geometry axis as transverse axis is realized using machine data:

MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function)

Example:

MD20100 $MC_DIAMETER_AX_DEF="X" ; geometry axis X is the transverse axis in the channel.

For this axis, diameter programming and assigning a constant cutting speed with G96/G961/G962 are both permitted.

Several transverse axes in the channel

The axis-specific machine data:
MD30460 BASE_FUNCTION_MASK (axis functions)
allows the definition of additional transverse axes, for which the functionality of the axis-specific diameter programming is permitted:

Bit	Value	Meaning
2	0	Axis-specific diameter programming is **not** permitted.
	1	Axis-specific diameter programming is permitted.

Note

The setting MD30460 bit 2 = 1 is only possible for linear axes.

An axis can be simultaneously defined in MD20100 and in MD30460 (bit 2). For this, the channel-specific MD20100 has a higher priority than the axis-specific MD30460.

With:

- MD20100, the function G96/G961/G962 is assigned to the transverse axis during power up.
- MD20100, the channel-specific diameter programming DIAMON, DIAMOF, DIAM90, DIAMCYCOF is assigned to the transverse axis during power up.

  After power up, this axis has the axis-specific basic position DIAMCHANA[AX].
- MD30460 bit2 the additional enabling of the axis-specific operations DIAMONA[AX], DIAMOFA[AX], DIAM90A[AX], DIACYCOFA[AX], DIMCHANA[AX].

Channel-specific basic position after power up, RESET

The channel-specific basic position after power up or RESET or end of parts program of the G group 29: DIAMON, DIAM90, DIAMOF, DIAMCYCOF define the

MD20150 $MC_GCODE_RESET_VALUE
and independently of
MD20110 $MC_RESET_MODE_MASK / bit0 the MD20152 $MC_GCODE_RESET_MODE.

The user can set the respective desired status via an event-controlled program call (prog-event).

If G96/G961/G962 is the basic position after power up, a transverse axis must be defined using MD20100 $MC_DIAMETER_AX_DEF, otherwise the alarm message 10870 is output.

Reference axis for G96/G961/G962 retained:

MD20110 $MC_RESET_MODE_MASK, bit 18=1 for RESET or end of parts program

MD20112 $MC_START_MODE_MASK, bit 18=1 for start of parts program

A reference axis for G96/G961/G962 can also be assigned without application of a transverse axis in MD20100 via SCC[AX]. For this scenario, the constant cutting speed cannot be activated with G96. For additional information, see:

Reference

Programming Manual Fundamentals, Feedrate Control and Spindle Motion
"Constant Cutting Speed (G96, G961, G962, G97, G971, LIMS, ACC[AX])"

11.3 Dimensional information for transverse axes

Transverse axes can be programmed with respect to both diameter and radius. Generally, they are diameter-related, i.e. programmed with doubled path dimension so that the corresponding dimensional information can be transferred to the part program directly from the technical drawings.

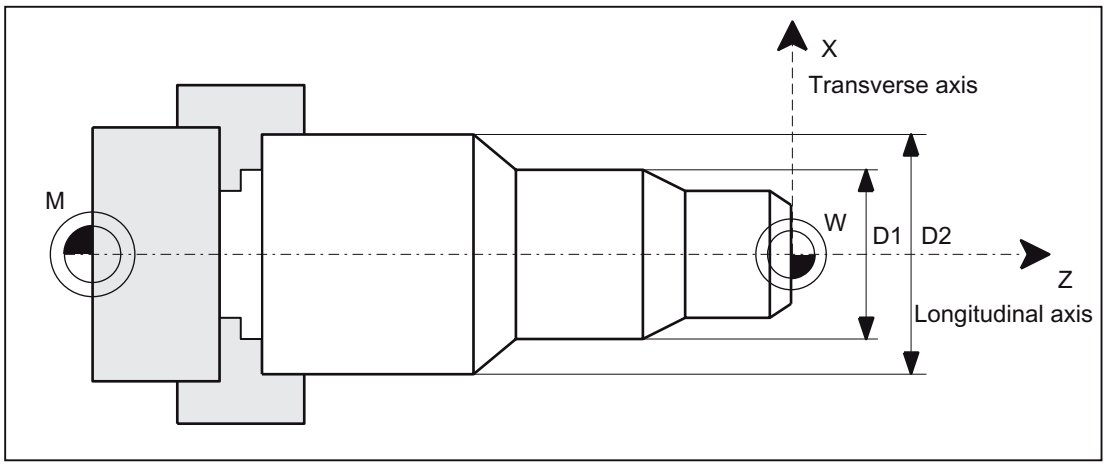


Figure 11-2 Transverse axis with diameter information (D1, D2)

Switching the diameter programming on/off

Channel-specific diameter programming

The activating or deactivating of the diameter programming is done via the modally active parts program statements of the G group 29:

- DIAMON: Diameter programming ON
- DIAMOF: Diameter programming OFF, in other words, radius programming ON
- DIAM90: Diameter or radius programming depending on the reference mode:
  - Diameter programming ON in connection with absolute dimensioning G90
  - Radius programming ON in connection with incremental dimensioning G91
- DIAMCYCOF: Radius programming for G90 and G91 ON , for the HMI, the last active G code of this group remains active

Reference is made exclusively to the transverse axis of the channel.

Axis-specific diameter programming for several transverse axes in one channel

Note

The additionally specified axis must be activated via MD30460 $MA_BASE_FUNCTION_MASK with bit2=1.

The axis specified must be a known axis in the channel. Geometry, channel or machine axes are permitted.
Programming is not permitted in synchronized actions.

The following axis-specific modal statements can be programmed several times in a parts program block:

- `DIAMONA[Axis]`: Diameter programming for `G90, G91 AC` and `IC` ON
- `DIAMOFA[Axis]`: Diameter programming OFF, in other words, radius programming ON
- `DIAM90A[axis]`: Diameter or radius programming depending on the reference mode:
  - Diameter programming ON in connection with absolute dimensioning `G90` and `AC`
  - Radius programming ON in connection with incremental dimensioning `G91` and `IC`
- `DIACYCOFA[axis]`: Radius programming for `G90` and `G91` ON , for the HMI, the last active G code of this group remains active
- `DIAMCHANA[axis]`: Acceptance of diameter programming channel status
- `DIAMCHAN`: all axes with MD30460, bit2=1 accept the diameter programming channel status

Axis-specific modal statements have priority over the channel setting.

Acceptance of the additional transverse axis in the channel

Due to a GET request from the parts program, the diameter programming status for an additional transverse axis is accepted in the new channel during axis replacement using RELEASE[axis].

Axis replacement in synchronized actions

For axis replacement in synchronized actions, a transverse axis takes the status of the axis-specific diameter programming with it into the new channel if the following applies to the transverse axis:

- with MD30460, bit2=1 axis-specific diameter programming is permitted.
- it is not subordinated to the channel-specific diameter programming in the releasing channel.

The active dimension can be queried via the system variable $AA_DIAM_STAT[AX].

Axis replacement via axis container rotation

By rotating the axis container, the assignment of a channel axis can change to assignment of a machine axis. The current diameter programming status is retained however for the channel axis after the rotation. This also applies to the current channel status and axis status, because the status is the same for all axes of the axis container at the time of the machine data "putting into effect" the status from MD30460 $MA_BASE_FUNCTION_MASK.

Initial setting

The following machine data is used to parameterize the initial setting:

MD20150 $MC_GCODE_RESET_VALUES [28] (initial setting of the G groups)
and independently of MD20110 $MC_RESET_MODE_MASK for bit0 the
MD20152 $MC_GCODE_RESET_MODE

Diameter-related data

After activation of the diameter programming, the following data refer to diameter dimensions:

DIAMON/DIAMONA[AX]

- Display data of transverse axis in the workpiece coordinate system:
  - Setpoint and actual position
  - Distance-to-go
  - REPOS Offset
- "JOG" mode:
  - Increments for incremental dimension (INC) and handwheel travel (dependent upon active MD)
- Part program programming:
  - End positions, independent of reference mode (`G90` / `G91`)
  - Interpolation parameters of circular-path programming (`G2` / `G3`) if these are programmed with part program instruction: `AC` absolute.
- Actual values read with reference to the workpiece coordinate system (WCS):
  - $AA_MW[*Transverse axis*]

    System variable of the measuring functions `MEAS` (measuring with delete distance-to-go) and `MEAW` (measuring without delete distance-to-go)
  - $P_EP[*Transverse axis*]
  - $AA_IW[*Transverse axis*]

DIAM90/DIAM90A[AX]

After activation of the reference-mode-dependent diameter programming, the following data are always displayed in relation to diameter regardless of the operating mode (G90 / G91):

- Actual value
- Actual values read with reference to the workpiece coordinate system (WCS):
  - $AA_MW[*Transverse axis*]

    System variable of the measuring functions MEAS (measuring with delete distance-to-go) and MEAW (measuring without delete distance-to-go)
  - $P_EP[*Transverse axis*]
  - $AA_IW[*Transverse axis*]

DIAMCYCOF/DIACYCOFA[AX]

Just as for DIAMCYCOF, a changeover to radius programing takes place within the controller for DIACYCOFA[AX]. The diameter programming status that was active before DIAMCYCOF or DIACYCOFA[AX] continues to be displayed to the HMI.

Permanently radius-related data

For transverse axes, the following data is **always** entered, programmed and displayed in relation to radius:

- Offsets:
  - Tool offsets
  - Programmable and configurable frames
  - External work offset
  - DRF and preset offset
  - etc.
- Working area limitation
- software limit switch
- Feed
- Display data with reference to the machine coordinate system
- Display data of the service images for axis, FSD and MSD

Extended functions for data that is always radius-related:

The following applies for PLC axes, via FC18 or axes controlled exclusively from the PLC:

- The dimension for PLC axes in the radius also applies to several transverse axes with diameter function and is independent of channel-specific or axis-specific diameter programming.
- In the JOG mode (Inc) a PLC axis is subordinate to the channel status. If diameter programming is active and MD20624 $MC_HANDWH_CHAN_STOP_COND bit 15 = 0, only half the path of the specified increment is traversed.

Radius programming from MD20100 $MC_DIAMETER_AX_DEF and MD30460 $MA_BASE_FUNCTION_MASK bit 2 is taken into account as follows depending on MD20360 $MC_TOOL_PARAMETER_DEF_MASK:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK		
Bit	**Value**	**Meaning**
3	0	Work offset $P_EXTFRAME and frames For transverse axes , work offsets in frames are always calculated as radius values.
5	0	External work office (axis overlay) For transverse axes, the external work offset is always calculated as radius value.
8	1	Display of remaining path in WCS always as a radius
9		For all transverse axes, with MD11346 $MN_HANDWH_TRUE_DISTANCE==1
	0	• half of the path of the specified handwheel increment is traveled, if channel-specific or axis-specific diameter programming is active for this axis.
	1	• half of the path of the specified handwheel increment is always traveled.
13	1	When jogging around circles, the circle center point coordinate is always a radius value, see SD42690 $SC_JOG_CIRCLE_CENTRE
14	1	For cycle masks, the absolute values of the transverse axis are in the radius.

Displaying position values in the diameter

Position values of the transverse axis are always displayed as a diameter value, if bit 0 = 1 is set by
MD27100 $MC_ABSBLOCK_FUNCTION_MASK.

Dimension on several transverse axes permanent diameter-related data

Several transverse axes permitted by MD30460 $MA_BASE_FUNCTION_MASK, bit 2 = 1 do not behave differently in comparison to a transverse axis defined using MD20100 $MC_DIAMETER_AX_DEF. Diameter values continue to be converted into radius values.

According to MD20360 $MC_TOOL_PARAMETER_DEF_MASK, for all of the transverse axes defined in the channel, the following functions can be activated as diameter:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK		
Bit	**Value**	**Meaning**
1	1	Transverse axis tool length as a diameter
2	1	Alarm for wear or tool length as a diameter and plane change
3	1	Work offset in frames of the transverse axis as a diameter
4	1	Preset value as a diameter
5	1	External work offset of transverse as a diameter
6	1	Actual values of the transverse axis as a diameter
7	1	Display of actual values of the transverse axis as a diameter value
10	1	Tool portion of an active tool carrier that can be oriented if no tool is active
11	1	Evaluation of $TC_DP6 as a diameter
12	1	Evaluation of $TC_DP15 as wear of the tool diameter
15	1	Incremental values of the transverse axis for cycle masks as diameter

Work offset $P_EXTFRAME and frames

Bit 3 = 1: For all transverse axes
, work offsets in frames are always calculated as diameter values. The frame stores the work offsets internally as a radius value. There is no conversion during a change of diameter, to radius programming or vice versa.

External work offset

Bit 5 = 1: For all transverse axes,
external work offsets are always calculated as diameter values. There is no conversion during a change of diameter, to radius programming or vice versa.

Settable response of geometry axes for traveling with handwheel

If the geometry axis is traveled as a transverse axis in the channel for handwheel traveling MD11346 $MN_HANDWH_TRUE_DISTANCE == 1, the response of the handwheel traveling can be changed via MD20624 $MC_HANDWH_CHAN_STOP_COND, bit15:

Bit 15 = 0: Only the half path of the specified increment is traveled.

Bit 15 = 1: The specified increment is traveled completely.

Application Examples

X is a transverse axis defined via MD20100 $MC_DIAMETER_AX_DEF.
Y is a geometry axis and U is an additional axis. These two axes are transverse axes with specified diameter further defined in MD30460 $MA_BASE_FUNCTION_MASK with bit2=1.
DIAMON is not active after power up.

```
N10 G0 G90 X100 Y50          ;no diameter programming is active
N20 DIAMON                   ;Channel-specific diameter programming, in effect for
                             X
N30 Y200 X200                ;Dimensions: X in the diameter, Y in the radius
N40 DIAMONA[Y]               ;axis-specific modal diameter programming,
                             ;in effect for Y
N50 Y250 X300                ;Dimensions: X and Y in diameter
N60 DIAM90                   ;Dimensions: X G90/AC in the diameter, G91/IC in the
                             radius
N70 Y200                     ;Y: continuing, axis-specific modal diameter
                             programming
N75 G91 Y20 U=DIC(40)        ;Dimensions: Y in the diameter, U non-modally IC in
                             the diameter
N80 X50 Y100                 ;Dimensions: X in the radius (G91), Y in the diameter
N85 G90 X100 U200            ;Dimensions: X in the diameter, U in the radius
N90 DIAMCHANA[Y]             ;Y accepts the channel status DIAM90
N95 G91 X100 Y100            ;Dimensions: X and Y in the radius(G91)
N100 G90 X200 Y200           ;Dimensions: X and Y in diameter
```

Example with axle replacement

Transverse axes with diameter specification applied as in the previous example.
X and Y are located in channel 1 and are also known in channel 2, i.e. permitted for axis replacement.

```
Channel 1
N10 G0 G90 X100 Y50       ;no diameter programming is active
N20 DIAMON                ;Channel-specific diameter programming for X
N30 Y200 X200             ;Dimensions: X in the diameter, Y in the radius
N40 DIAMONA[Y]            ;Y axis-specific modal diameter programming
N50 Y250 X300             ;Dimensions: X and Y in diameter
N60 SETM(1)               ;Synchronous marker 1
N70 WAIT(1,2)             ;wait for synchronous marker 1 in channel 2
Channel 2
...
N50 DIAMOF                ;channel 2 no diameter programming active
...
N100 WAIT(1,1)            ;wait for synchronous marker 1 in channel 1
N110 GETD(Y)              ;Axis replacement direct Y
N120 Y100                 ;Y the channel-specific diameter programming
                          ;subordinated in channel 2, i.e. dimension in the radius
```

11.4 Data lists

11.4.1 Machine data

11.4.1.1 Channelspecific machine data

Number	Identifier: $MC_	Description
20050	AXCONF_GEOAX_ASSIGN_TAB[n]	Assignment of geometry axis to channel axis
20060	AXCONF_GEOAX_NAME_TAB[n]	Geometry axis name in channel
20100	DIAMETER_AX_DEF	Geometry axis with transverse axis function
20110	RESET_MODE_MASK	Definition of control basic setting after powerup and RESET / part program end
20112	START_MODE_MASK	Definition of the control basic settings for NC start
20150	GCODE_RESET_VALUES[n]	Reset G groups
20152	GCODE_RESET_MODE[n]	G code basic setting at RESET/end of parts program
20360	TOOL_PARAMETER_DEF_MASK	Definition of tool parameters
20624	HANDWH_CHAN_STOP_COND	Definition of the behavior of traveling with handwheel
27100	ABSBLOCK_FUNCTION_MASK	Parameterize block display with absolute values

11.4.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30460	BASE_FUNCTION_MASK	Axis functions

P3: Basic PLC program for SINUMERIK 840D sl 12

12.1 Brief description

General

The PLC basic program organizes the exchange of signals and data between the PLC user program and the NCK (Numerical Control Kernel), HMI (Human Machine Interface) and MCP (Machine Control Panel). In the case of signals and data, a distinction is made between the following groups:

- Cyclic signal exchange
- Event-driven signal exchange
- Messages

Cyclic signal exchange

The cyclically-exchanged signals consist primarily of bit arrays.

- They contain **commands** transferred from the PLC to the NCK (such as start or stop) and **status information** from the NCK (such as program running, interrupted, etc.).
- The bit fields are organized into signals for:
  - Mode group
  - Channels
  - Axes/spindles
  - General NCK signals

The cyclic exchange of data is performed by the basic program at the start of the PLC cycle (OB1). This ensures, for example, that the signals from the NCK remain constant throughout a cycle.

Event-driven signal exchange NCK → PLC

PLC functions that have to be executed as a function of the workpiece program are triggered by auxiliary functions in the workpiece program. If a block with auxiliary functions is executed, the type of auxiliary function determines whether the NCK has to wait for this function to execute (e.g. tool change) or whether the function will be executed together with the workpiece machining process (e.g. tool loading on milling machines with chain magazine).

Data transfer must be as fast and yet as reliable as possible, in order to minimize the effect on the NCK machining process. Data transfer is, therefore, interrupt- and acknowledgement-driven. The basic program evaluates the signals and data, acknowledges this to the NCK and transfers the data to the application interface at the start of the cycle. If the data does not require user acknowledgement, this does not affect NC processing.

Event-driven signal exchange PLC → NCK

An "event driven signal exchange PLC → NCK" takes place whenever the PLC transfers a request to the NCK (e.g. traversal of an auxiliary axis). In this case, data transfer is also acknowledgement-driven. When performed from the user program, this type of signal exchange is triggered using a function block (FB) or function call (FC).

The associated FBs (Function Blocks) and FCs (Function Calls) are supplied together with the basic program.

Messages

User messages are acquired and conditioned by the basic program. The message signals are transferred to the basic program via a specified bit array. where they are evaluated and, if message events occur, entered in the PLC's interrupt buffer by means of the ALARM S/SQ functions. If an HMI (e.g. SINUMERIK Operate) is being used, the messages are transferred to the HMI and displayed.

PLC/HMI data exchange

In this type of data exchange, the HMI takes the initiative, being referred to as the "client" on the bus system. The HMI polls or writes data. The PLC processes these requests at the cycle control point via the operating system. The PLC basic program is not involved in these exchanges.

Note

The function of the machine is largely determined by the PLC program. Every PLC program in the RAM can be edited with the programming device.

Know-how protection for user blocks

To protect the know-how contained in the the user blocks (OB, FB and FC), they can be encoded with the SBP tool (SIMATIC block protection) contained in SIMATIC STEP 7. These blocks can then no longer be opened, debugged and modified without specifying the password for the encoding.

When encoding, the automation system on whose PLC-CPU the blocks are to be executed, must be specified: SIMATIC and/or SINUMERIK PLC-CPU.

The handling of the blocks, e.g. loading to the CPU, is not affected by the encoding.

Requirement

SIMATIC STEP 7 as of Version 5.5 SP3

12.2 Key data of the PLC CPU

Key data of the PLC CPU

The overview of the key data of the PLC CPU integrated in the SINUMERIK NCU can be found in:

References

NCU 7x0.3 PN Manual, Section "Technical data"

NOTICE
I/O addresses for integrated drives The I/O addresses above 4096 are reserved for the integrated drives of the NCU and must not assigned otherwise.

Functions of the basic PLC program

Scope	
Axes/spindles	31
Channels	10
Mode groups	10
Functions	
Status/control signals	+
M decoders (M00-99)	+
G group decoders	+
Aux. function distributors	+
Aux. function transfer, interrupt-driven	+
M decoding acc. to list	+
Move axes/spindles from PLC	+
ASUB interface	+
Error/operating messages	+
Transfer MCP and HHU signals	+
Display control handheld unit	+
Read/write NCK variables and GUD	+
PI services	+
Tool management	+
Star/delta switchover	+
m:n	+
Safety Integrated	+
Program diagnostics	+

12.3 PLC operating system version

The PLC operating system version is displayed at:

- User interface of SINUMERIK Operate: "Operating area switchover" > "Diagnostics" > "Version" ⇒ version data / system software NCU: Selection "PLC" > "Details" ⇒ version data / system software NCU/PLC: The PLC operating system version is displayed in the first line is at "PLC 3xx…".
  Note
  The displayed version is SINUMERIK-specific. It is not compatible with the basic SIMATIC CPU.
- SIMATIC STEP 7, HW Config: In the properties of the PLC CPU in the SINUMERIK rack: "Properties - CPU 3xx…" > "Order no. / firmware": xxxx / **Vx.y.z**
  Note
  The version of the basic SIMATIC CPU is displayed.

12.4 PLC mode selector

The PLC mode selector is located on the front of the NCU module. The following PLC operating modes can be set via the PLC mode selector:

S [1)]	Meaning	Remark
0	RUN-P	The PLC program can be changed without activation of the password
1	RUN	Only read access operations are possible using a programming device (PG). It is not possible to make changes to the PLC program until the password has been set.
2	STOP	Processing the PLC program is stopped and all PLC outputs are set to substitute values.
3	MRES	The PLC is switched into the STOP state followed by a PLC general reset (default data).
1) Switch position of the PLC mode selector		

References

A detailed description of the position of the PLC mode selector on the front of the NCU module, as well as its use in connection with NCK and PLC general reset can be found in:

CNC Commissioning Manual: NCK, PLC, Drive:

- Section "Switch-on/power-up" > "Operator control and display elements for power-up"
- Section "Switch-on/power-up" > "NCK and PLC general reset"
- Section "General tips" > "Separate NCK and PLC general reset"

12.5 Reserve resources (timers, counters, FC, FB, DB, I/O)

Reserve resources (timers, counters, FC, FB, DB, I/O)

The components below are reserved for the basic program:

- **Timers**

  No reservation

- **Counter**

  No reservation

- **FC, FB, DB**

  FC 0 to FC 29 and FB 0 to FB 29 are reserved for the basic program. The number range between 1000 and 1023 is also reserved for FCs and FBs. Data blocks DB 1 to DB 62 and DB 71 to DB 80 are reserved. The number range 1000 to 1099 is also reserved in addition for DB. The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as desired by the user.

- **I/O range**

  The PLC has an I/O address volume of 8192 bytes each for inputs and outputs. The address ranges starting at 4096 / 4096 are reserved for/occupied by integrated drives. However, diagnostic addresses for modules can be assigned to the highest address range as proposed by STEP 7. Furthermore, the address range between 256 and 287 is assigned for the NCK, CP and HMI in rack 0 on the SIMATIC 300 station.

12.6 Commissioning hardware configuration of the PLC CPU

The commissioning of the PLC CPU is described in detail in:

References

CNC Commissioning Manual: NCK, PLC, Drive:

- Section: "Connect PG/PC to PLC"
- Section: "Commissioning PLC"
- Section: "Basics" > "PLC program"
- Section: "General tips" > "Separate NCK and PLC general reset"
- Section: "General tips" > "Integrating PG/PC into the network (NetPro)"

12.7 Starting up the PLC program

12.7.1 Installation of the basic program

The installation of the basic program is described in detail in:

References

CNC Commissioning Manual: NCK, PLC, Drive; Section: "Commissioning PLC" > "Creating a PLC program"

Note

Installation/update

Before installing the toolbox for SINUMERIK 840D sl, SIMATIC STEP 7 must be installed.

It is recommended that the hardware expansions for STEP 7 be installed again from the toolbox after an update of STEP 7.

Contents

The OB source programs, including standard parameterization, interface symbols and data-block templates for the handheld unit and M decoding functions are included in the basic program.

12.7.2 Application of the basic program

A new CPU program (e.g. "Turnma1") must be set up in a project by means of the STEP 7 software for each installation (machine).

Remark

The catalog structures of a project and the procedure for creating projects and user programs are described in the relevant SIMATIC documentation.

Procedure

The basic program blocks are copied using the SIMATIC Manager and "File" > "Open" > "Library".

The following components must be copied from the library:

- From the block container: FCs, FBs, DBs, OBs, SFC, SFB, UDT
- The source_files (from the source container): GPOB840D
- Possibly MDECLIST, HHU_DB and others
- The symbols table (from the symbols container)

Compatibility with STEP 7

There are no dependencies between the basic program and current STEP 7 versions.

12.7.3 Version codes

Basic program

The version of the basic program is displayed on the Version screen of the user interface along with the controller type.

The controller type is encoded as follows:

Left-justified decade of DB 17 DBD 0 (byte 0)	Controller type
03	SINUMERIK 840D sl (NCU 7x0)

User program

Users can also display their own PLC version codes in the Version screen. For this purpose, a data of type STRING containing a maximum of 54 characters must be defined in any data block. The data can contain a text of the user's choice. Parameterizations for this string are made via a pointer in FB 1. Parameterization requires symbolic definition of the data block.

See also section "block descriptions" > "FB 1: RUN_UP Basic program, startup section (Page 999) ".

12.7.4 Machine program

The machine manufacturer creates the machine program using the library routines supplied with the basic program. The machine program contains the logic operations and sequences on the machine. The interface signals to the NC are also controlled in this program. More complex communication functions with the NCK, e.g. read/write NC data, tool-management acknowledgments, etc., are activated and executed via blocks FCs and FBs of the basic-program).

The machine program can be created in various STEP 7 creation languages, e.g. AWL, KOP, FUP, S7-HIGRAPH, S7GRAPH, SCL. The complete machine program must be generated and compiled in the correct sequence.

This means that blocks that are called by other blocks must generally be compiled **before these** blocks.

If blocks that are called by other blocks are subsequently modified in the interface (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) as the program is developed, then the call block and all blocks associated with it must be compiled again. This general procedure applies analogously to instance data blocks for FBs. If these sequence of operations is not maintained, time-stamp conflicts occur when the data retranslated into STEP 7. As such, the recompilability of the blocks is not ensured and with the function "Status of block" unnecesary conflicts can also appear. It is, moreover, advisable to generate blocks in ASCII-STL by means of the STEP 7 editor when they have been created in Ladder Diagram or in single statements (incremental mode).

12.7.5 Data backup

The PLC-CPU does not save any symbolic names, but instead only the datatype descriptions of the block parameters VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR and the datatypes of the global data blocks.

Note

No sensible recompilation is possible without the related project for this machine. This especially affects, for instance the function status of the block or the necessary changes done in the PLC-CPU programs later. It is, therefore, necessary to keep a backup copy of the STEP 7 project located in the PLC CPU on the machine. This is a great help for the service case and saves unnecessary consumption of time in restoring the original project.

If the STEP 7 project exists and has been created according to the instructions given above, then symbols can be processed in the PLCCPU on this machine. It may also be advisable to store the machine source programs as ".awl" files in case they are required for any future upgrade.

The source programs of all organization blocks and all instance data blocks should always be available.

12.7.6 PLC series startup, PLC archive

Once the blocks have been loaded to the PLC CPU, a series archive can be generated via the HMI user interface to back up data on the machine. To ensure data consistency, this backup must be created immediately after block loading when the PLC is in the Stop state. It does not replace the SIMATIC project backup as the series archive saves binary data only. For instance, no symbolic information is present here. In addition, no CPU DBs (SFC 22 DBs) or SDBs generated in the CPU are saved.

Selection of the SINUMERIK archiving program

The PLC series archive can be generated directly from the SIMATIC project as an alternative.

- Open the "Settings" dialog box in the SIMATIC Manager: Menu bar "Tools" > "Settings"
- Open the "Archive" tab
- Select the the SINUMERIK archiving program "SINUMERIK (*.arc)" in the "Preferred archiving program" drop-down list box.

Start of the SINUMERIK archiving program

The SINUMERIK archiving program is started in the SIMATIC Manager via the menu command "File" > "Archive".

The PLC archive is generated after assigning the archive name. If a project contains several program paths, the S7 program for which the PLC archive will be created can be selected in the dialog box. All blocks contained in the selected program path are archived (except data blocks created with SFC 22 (online) in the CPU).

The "SDB archive" function can be activated or deactivated for the archiving program. If "SDB archive" is activated, a PLC archive is created that only contains the system data blocks (SDB) of the selected program path.

Automation

The process of generating a series archive can be automated (comparable to the command interface in STEP 7). In generating this series archive, the command interface is expanded.

The following functions are available for this expansion:

The functions (shown here in VB script) are not available until server instantiations and Magic have been called:

```
Const S7BlockContainer = 1138689, S7PlanContainer = 17829889

Const S7SourceContainer = 1122308

set S7 = CreateObject("Simatic.Simatic.1")

instantiate rem command interface of STEP 7

Set S7Ext = CreateObject("SimaticExt.S7ContainerExt")

Call S7Ext.Magic("")
```

Functions:

- `Function Magic(bstrVal As String) As Long`
- `Function MakeSerienIB (FileName As String, Option As Long, Container As S7Container) As Long`

Description

Function **Magic**(bstrVal As String) As Long

The call provides access to certain functions. The function must be called once after server instantiation. The value of bstrVal can be empty. This initiates a check of the correct STEP 7 version and path name in Autoexec. The functions are enabled with a return parameter of 0.

Return parameter (-1) = incorrect STEP 7 version

Return parameter (-2) = no entry in Autoexec.bat

Function **MakeSeriesstart-up**(FileName As String, Option As Long, Container As S7Container) As Long

"Option" parameter:

0:	Normal series startup file with general reset
Bit 0 = 1:	Series startup file without general reset. When project contains SDBs, this option is inoperative. A general reset is then always executed
Bit 1 = 1:	Series startup file with PLC restart

Return parameter value:

0	= OK
-1	= Function unavailable, call Magic function beforehand
-2	= File name cannot be generated
-4	= Container parameter invalid or container block empty
-5	= Internal error (memory request rejected by Windows)
-6	= Internal error (problem in STEP 7 project)
-7	= Write error when generating series startup files (e.g. diskette full)

Use in script

Program code

```
If S7Ext.Magic("") < 0 Then
  Wscript.Quit(1)
End If
  Set Proj1 = s7.Projects("new")
  set S7Prog = Nothing
  Set s7prog = Proj1.Programs.Item(1) 'if there is only one program'
For Each cont In s7prog.Next
  If (Cont.ConcreteType = S7BlockContainer) Then
  ' Check block container
  Exit For
  End if
  Cont = Nothing
  Next
Error = S7Ext.MakeSerienIB("f:\dh\arc.dir\PLC.arc", 0, Cont)
' Now error analysis
```

The `For Each` ... `Next` block programmed above can be programmed in the Delphi programming language as follows (the programming for C, C++ programming languages is similar):

Program code

```
Var
  EnumVar: IEnumVariant;
  rgvar: OleVariant;
  fetched: Cardinal;

//For Each Next
EnumVar := (S7Prog.Next._NewEnum) as IEnumVariant;
While (EnumVar.Next(1,rgvar,fetched) = S_OK) Do Begin
  Cont := IS7Container(IDispatch(rgvar)); // block container
  Check sources
  If (Cont.ConcreteType = S7BlockContainer) Then Break;
  Cont := NIL;
End;
```

12.7.7 Software upgrade

A general PLC reset should be performed to achieve a defined initial state before the PLC software is upgraded. In this case, among other things, all user data (program and data blocks) will be deleted. The PLC general reset is described in:

References:
Commissioning Manual CNC: NCK, PLC, Drive, General Tips,
Section: PLC general reset

Generating a new SIMATIC S7 project

In normal cases, the new PLC basic program is to be linked-in for a new NCU software version. The basic programs blocks must be loaded into the user project for this purpose. If the following program and data blocks are already in the user project, then these should not be transferred with the blocks of the basic PLC program: OB 1, OB 40, OB 82, OB 86, OB 100, FC 12 and DB 4. These may have been modified by the user and should not be overwritten. The new basic program must be linked with the user program. The following procedure must be taken into account:

1. Generate the text or source file of all user blocks before copying the basic PLC program.
2. Copy the new basic program blocks into the SIMATIC S7 project (for a description, see Section "Application of the basic program (Page 914)")
3. All user programs "*.awl" must be recompiled in the correct order! (see also: " Machine program (Page 916)"):
4. This newly compiled SIMATIC S7 project should then be downloaded with STEP 7 into the PLC.

However, it is normally sufficient to recompile the organization blocks (OBs) and the instance data blocks of the S7 project. This means before upgrading, only the sources for the organization blocks and the instance data blocks have to be generated.

NC variables

The latest NC VAR selector can be used for each NC software version (even earlier versions). The variables can also be selected from the latest list for earlier NC software versions. The data content in DB 120 (default DB for variables) does not depend on the software status. That is, variables selected in an older software version need not be reselected when the software is upgraded.

12.7.8 I/O modules (FM, CP modules)

Additional packages for STEP 7 are generally required for more complex I/O modules (FM, CP modules). Support blocks (FC/FB) are provided in these additional packets. The blocks contain specific functions for operating the relevant module. These functions can be parameterized and called in the user program.

Identical numbers

If handling and basic program blocks have identical numbers, the block numbers of the basic program must remain unchanged. The block numbers of the handling blocks must be renamed to free numbers via STEP 7.

12.7.9 Troubleshooting

This section describes problems which may occur, their causes and remedies and should be read carefully before hardware is replaced.

Errors, cause/description and remedy			
Serial no. error information	**Errors**	**Cause/description**	**To correct or avoid errors**
1	No connection via MPI to PLC.	The MPI cable is not plugged in or is defective. Possibly, the STEP 7 software is also not correctly configured for the MPI card.	Test: Create a link with the programmer in the STEP 7 editor by means of connection "Direct_PLC". A number of node addresses must be displayed here. If they do not appear, the MPI cable is defective/not plugged in.
2	PLC cannot be accessed in spite of PLC general reset.	A system data block SDB 0 has been loaded with a modified MPI address. This has caused an MPI bus conflict due to dual assignment of addresses.	Disconnect all MPI cables to other components. Create the link "Direct_PLC" with the programmer. Correct the MPI address.
3	All four LEDs on the PLC flash (DI disaster)	A system error has occurred in the PLC. **Measures:** The diagnostic buffer on the PLC must be read to analyze the system error in detail. To access the buffer, the PLC must be stopped (e.g. set "PLC" switch to position 2). A hardware reset must then be performed. The diagnostic buffer can then be read out with STEP 7. Relay the information from the diagnostic buffer to the Hotline / Development Service. A general reset must be carried out if requested after the hardware RESET. The diagnostic buffer can then be read with the PLC in the Stop state.	Once the PLC program has been RESET or reloaded, the system may return to normal operation. Even in this case, the content of the diagnostic buffer should be sent to the Development Office.

12.8 Coupling of the PLC CPU

12.8.1 General

A CPU of the S7-300 automation system is used as PLC for the SINUMERIK 840D sl. The PLC-CPU is integrated into the NCU component as a sub-module. A reference to the performance data of the PLC CPU can be found in Section "Key data of the PLC CPU (Page 909)".

12.8.2 Properties of the PLC CPU

The PLC integrated in the SINUMERIK 840D sl generally has the same functionality as the corresponding SIMATIC S7-300 PLC.

For differences, see reference in Section "Key data of the PLC CPU (Page 909)".

Owing to differences in their memory system as compared to a SIMATIC S7-300 PLC, certain functions are not available (e.g. save blocks on memory card, save project on memory card).

Note

As with the PLC integrated in SINUMERIK, there is no automatic start of the PLC after power failure and recovery for a SIMATIC S7-300 PLC when a "PLC stop" is triggered by an operator action on the programming device. For safety reasons, the PLC remains in the stop state with an appropriate diagnostic entry. You can start the PLC only by an operator action on the programming device, "Execute a restart", or via the mode selector "Stop" > "Run" (warm restart).

12.8.3 Interface with integrated PLC

Physical interfaces

With the SINUMERIK 840D sl, the PLC integrated in the NCU offers the option of exchanging signals between the NCK and PLC directly via a dual-port RAM.

Data exchange with the operator panel

Data exchange with the operator panel (e.g. TCU/OP) can be performed via Ethernet or PROFIBUS. With a connection via Ethernet, communication takes place via the integrated communication processor (CP 840D sl).

Data exchange with the machine control panel (MCP) and handheld unit (HHU) can be performed via MPI, PROFIBUS or Ethernet.

Programming devices should preferably be connected via Ethernet or via MPI (Multi-Point Interface) directly to the PLC.

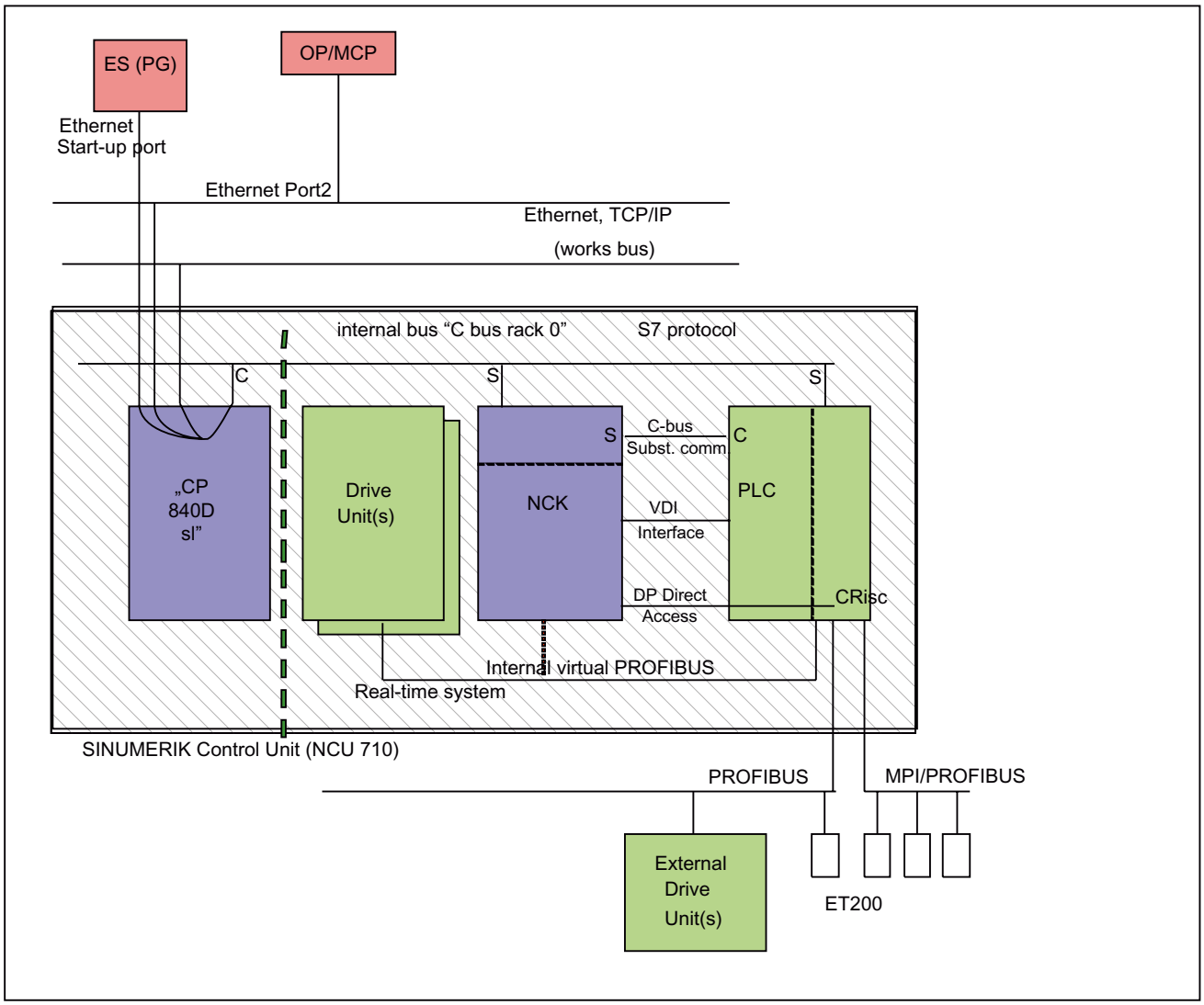


Figure 12-1 NCK/PLC coupling on SINUMERIK 840D sl (integrated PLC)

Interface: NCK/PLC

The data exchange between NCK and PLC is organized by the basic program on the PLC side. The **status information**, such as "Program running", stored by the NCK in the NCK/PLC interface is copied to data blocks by the basic program at the beginning of the cycle (OB 1) and can then be accessed in the user program (user interface). The **control signals** for the NCK (e.g. NC start) entered in the interface data block by the user are also written to the internal DPR and transferred to the NCK at the start of the cycle.

Workpiece-program-specific **auxiliary functions** transferred to the PLC are first evaluated by the basic program (interrupt-driven) and then transferred to the user interface at the start of OB 1. If the relevant NC block contains auxiliary functions that require that NCK processing is interrupted (e.g. M06 for tool change), the basic program stops the decoding of the NCK block initially for one PLC cycle. The user can then use the "read disable" interface signal to halt the block execution until the tool change has been completed. If, on the other hand, the relevant NC block only contains auxiliary functions, which do not require interruption of the decoding (e.g. M08 for cooling medium on), the transfer of these "fast" auxiliary functions is directly acknowledged in OB 40, so that decoding is only insignificantly influenced by the transfer to the PLC.

The evaluation and enabling of the **G functions** transferred from the NCK are also alarm-driven, however they are transferred directly to the user interface. Where a G function is evaluated at several points in the PLC program, differences in the information of the G function within one PLC cycle may arise.

In the case of **NC actions** triggered and assigned with parameters by the PLC (e.g. traverse concurrent axes), triggering and parameter assignment is performed using FCs and FBs, not interface data blocks. The FCs and FBs belonging to the actions are supplied together with the basic program. The FCs and FBs required must be loaded by the user and called in the PLC program of the machine manufacturer (machine program). For an overview of FC, FB and data blocks, sorted according to basic and extended functions, please refer to Section "Start-up of PLC programs".

Interface: HMI/PLC

HMI/PLC data exchange is performed via the integrated CP, whereby the HMI is always the active partner (client) and the PLC is always the passive partner (server). Data transferred or requested by the HMI is read from and written to the HMI/PLC interface area by the PLC operating system (timing: Cycle control point). From the viewpoint of the PLC application, the data is identical to I/O signals.

Interface: MCP/PLC or HHU/PLC (connection: Ethernet)

MCP/PLC and HHU (HT2) / PLC data exchange is performed via the integrated CP. The CP transfers the MCP/HHU signals to and fetches them from the PLC's internal DPR (Dual-Port RAM). On the PLC side, the basic program handles communication with the user interface. The basic program parameters define the operand areas (e.g. I/O areas) and the start addresses via the parameters of the basic programs (FB 1, DB 7).

Interface: MCP/PLC (connection: PROFIBUS)

MCP/PLC data exchange takes place via the PLC's PROFIBUS. The MCP's I/O addresses are to be set in the PLC's process image area and via HW configuration in STEP 7. The MCP*In, MCP*Out pointer variables must be set to the same addresses. The selected DP slave number must be entered in MCP*BusAdr.

Interface: HHU/PLC (connection: MPI)

The HHU/PLC data exchange is performed via the MPI interface on the PLC. The "Communication with global data (GD)" service is used for this purpose (see also STEP 7 User Manual). The PLC operating system handles the transfer of signals from and to the user interface. The STEP 7 "Communication configuration" configuring tool is used to define both GD parameters as well as operand areas (e.g. I/O areas) and their start addresses.

12.8.4 Diagnostic buffer on PLC

The diagnostic buffer of the PLC (readable using STEP 7) will enter diagnostic information on the PLC operating system.

12.9 Interface structure

Interface DBs

Mapping in interface data blocks is necessary due to the large number of signals exchanged between the NCK and PLC. These are global data blocks from the viewpoint of the PLC program. During system start-up, the basic program creates these data blocks from current NCK machine data (no. of channels, axes, etc.). The advantage of this approach is that the minimum amount of PLC RAM required for the current machine configuration is used.

12.9.1 PLC/NCK interface

General

The PLC/NCK interface comprises a data interface on one side and a function interface on the other. The data interface contains status and control signals, auxiliary functions and G functions, while the function interface is used to transfer jobs from the PLC to the NCK.

Data interface

The data interface is subdivided into the following groups:

- NCK-specific signals
- Mode-group-specific signals
- Channel-specific signals
- Axis/spindle/drive-specific signals

Function interface

The function interface is formed by FBs and FCs. The figure below illustrates the general structure of the interface between the PLC and the NCK.

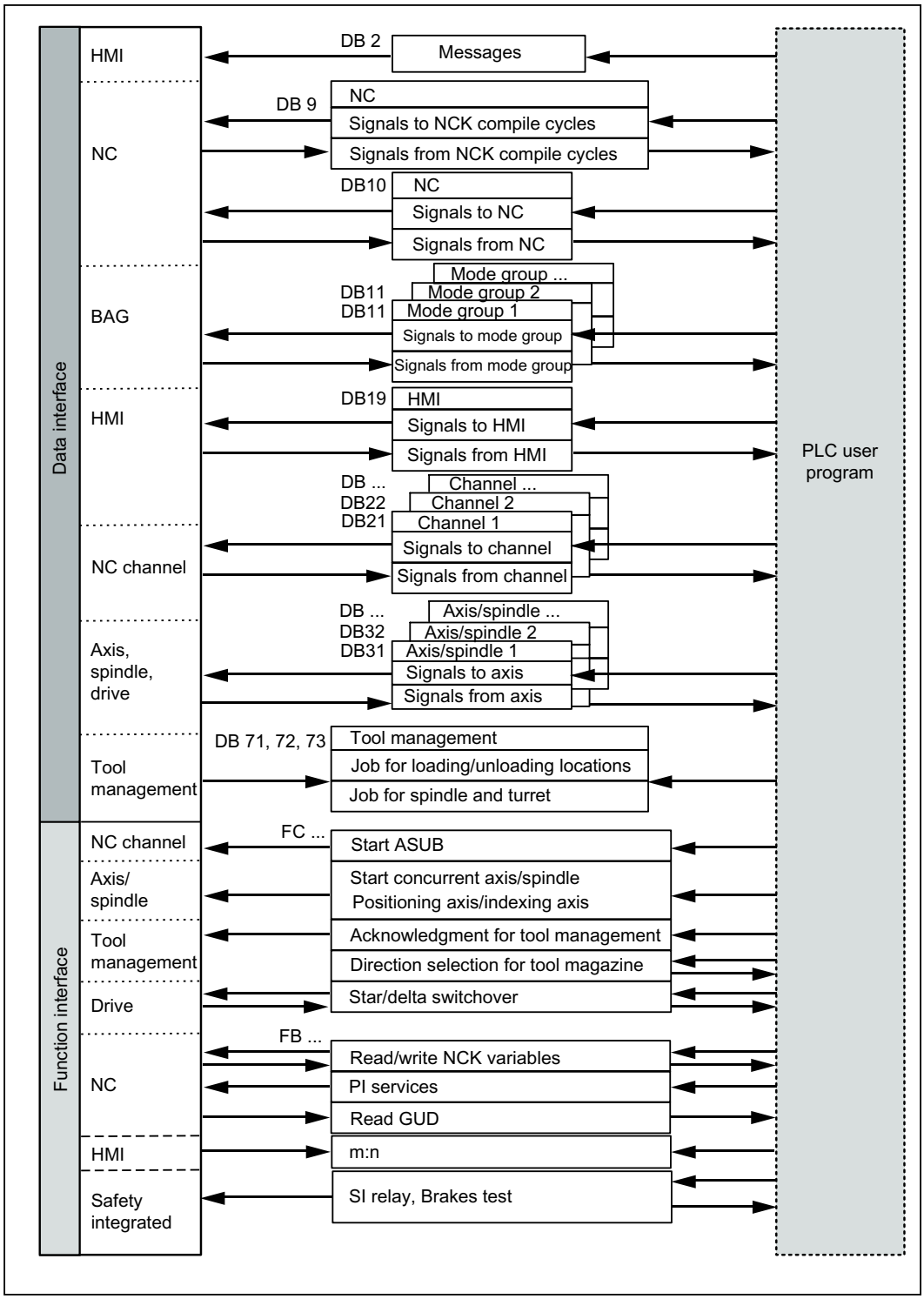


Figure 12-2 PLC/NCK user interface

Compile-cycle signals

In addition to the standard signals exchanged between the PLC and NCK, an interface data block for compile cycles is also generated if required (DB 9). The associated signals which are dependent on the compile cycles are transmitted cyclically at the start of OB 1. The basic program starts transmission at the lowest address and works up to the highest. First, signals are transferred from the PLC to the NCK, then from the NCK to the PLC. The user must synchronize the NCK and PLC as necessary (e.g. using the semaphore technique). Signal transmission is asynchronous between NCK and PLC. This means, for example, that active NCK data transmission can be interrupted by the PLC. This can mean that data is not always consistent.

PLC/NCK signals

The group of signals from the PLC to NCK includes:

- Signals for modifying the digital and analog I/O signals of the NCK
- Keyswitch and emergency stop signals

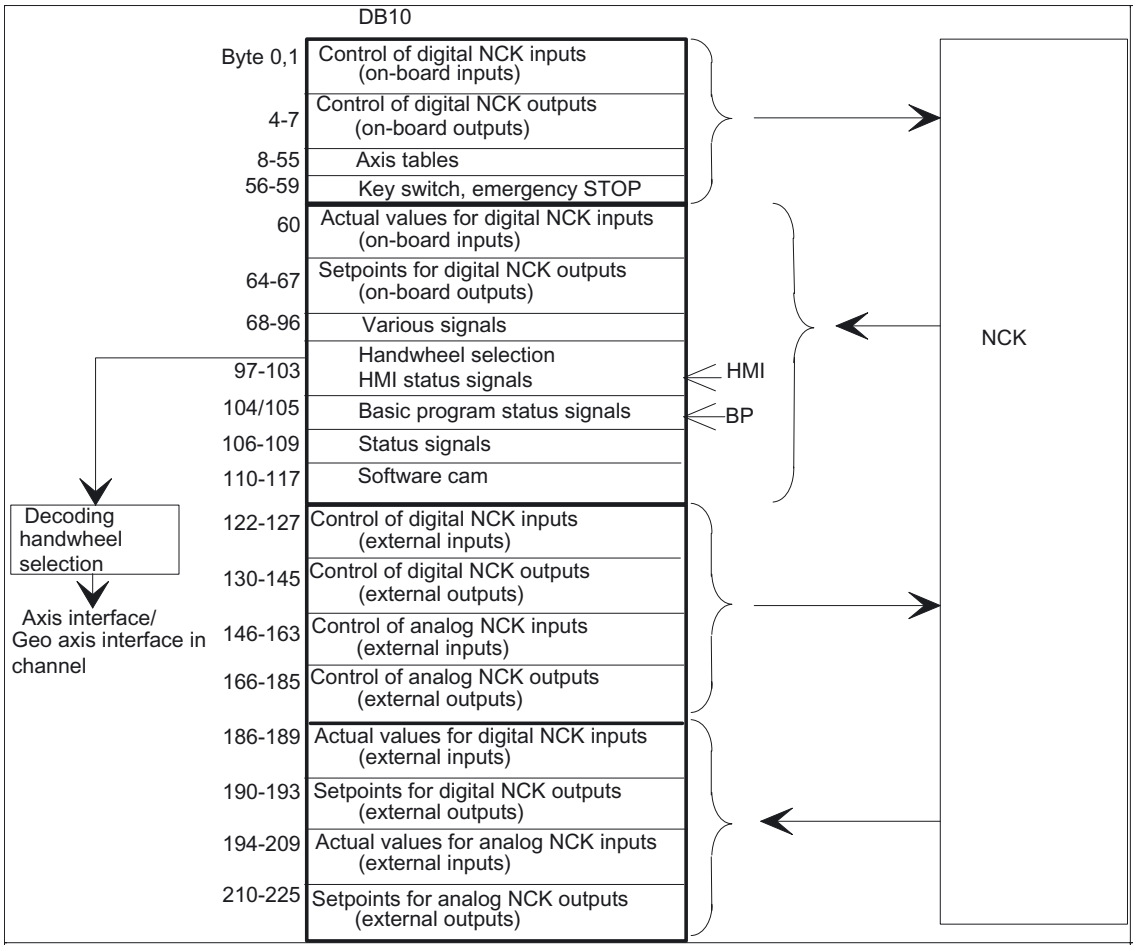


Figure 12-3 PLC/NCK interface

NCK/PLC signals

The group of signals from the NCK to PLC includes:

- Actual values of the digital and analog I/O signals of the NCK
- Ready and status signals of the NCK

Also output in this group are the HMI handwheel selection signals and the status signals.

The signals for handwheel selection are decoded by the basic program and entered in the machine/axis-specific interface.

Digital/analog I/Os of the NCK

The following must be noted with respect to the digital and analog I/Os of the NCK:

Inputs:

- All input signals or input values of the NCK are also transferred to the PLC.
- The transfer of signals to the NC part program can be suppressed by the PLC. Instead, a signal or value can be specified by the PLC.
- The PLC can also transfer a signal or value to the NCK even if there is no hardware for this channel on the NCK side.

Outputs:

- All signals or values to be output are also transferred to the PLC.
- The NCK can also transfer signals or values to the PLC even if there is no hardware for this channel on the NCK side.
- The values transferred by the NCK can be overwritten by the PLC.
- Signals and values from the PLC can also be output directly via the NCK I/O devices.

Note

While realizing the digital and analog NCK I/Os the information contained in the following documentation must be taken into account:
References:
Functions Manual Extended Functions; Digital and analog NCK I/Os (A4)

Signals PLC/Mode group

The operating mode signals set by the machine control panel or the HMI are transferred to the operating mode group (BAG) of the NCK. These apply to all NCK channels. Several mode groups can be optionally defined in the NCK.

The mode group reports its current status to the PLC.

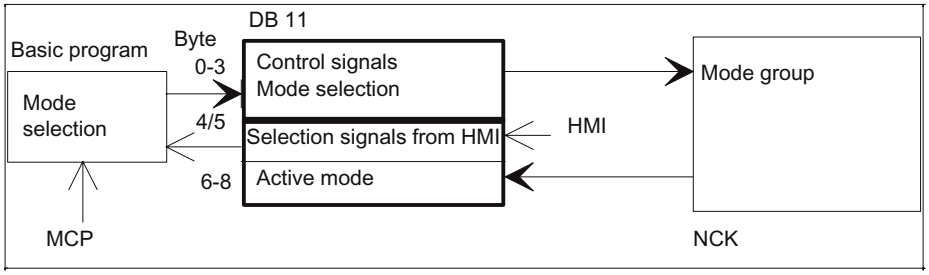


Figure 12-4 PLC/Mode group interface

Signals PLC/NCK channels

The signal groups below must be considered on the interface:

- Control/status signals
- Auxiliary/G functions
- Tool management signals
- NCK functions

The **control/status functions** are transmitted cyclically at the start of OB1. The signals entered in the channel-specific interface by the HMI (HMI signals are entered by the PLC operating system) are also transferred at this time if they have been defined on the HMI operator panel, not on the MCP.

Auxiliary functions and G functions are entered in the interface data blocks in two ways. First, they are entered with the change signals.

- The **M signals** M00 - M99 (they are transferred from the NCK with extended address 0) are also decoded and the associated interface bits set for the duration of one cycle.
- For **G functions**, only the groups selected via machine data are entered in the interface data block.
- The **S values** are also entered together with the related M signals (M03, M04, M05) in the spindle-specific interface. The axis-specific feedrates are also entered in the appropriate axisspecific interface.

When the **tool management (magazine management)** function is activated in the NCK, the assignment of spindle or revolver and the loading/unloading points are entered in separate interface DBs (DB71 - 73)

The triggering and parameter assignment of **NCK functions** is performed by means of PLC function calls.
The following function calls are available:

- Position a linear axis or rotary axis
- Position an indexing axis
- Start a prepared asynchronous subprogram (ASUB)
- Reading/writing of NC variables
- Update magazine and tool motion

Some of the above functions are described in their own function documentation.

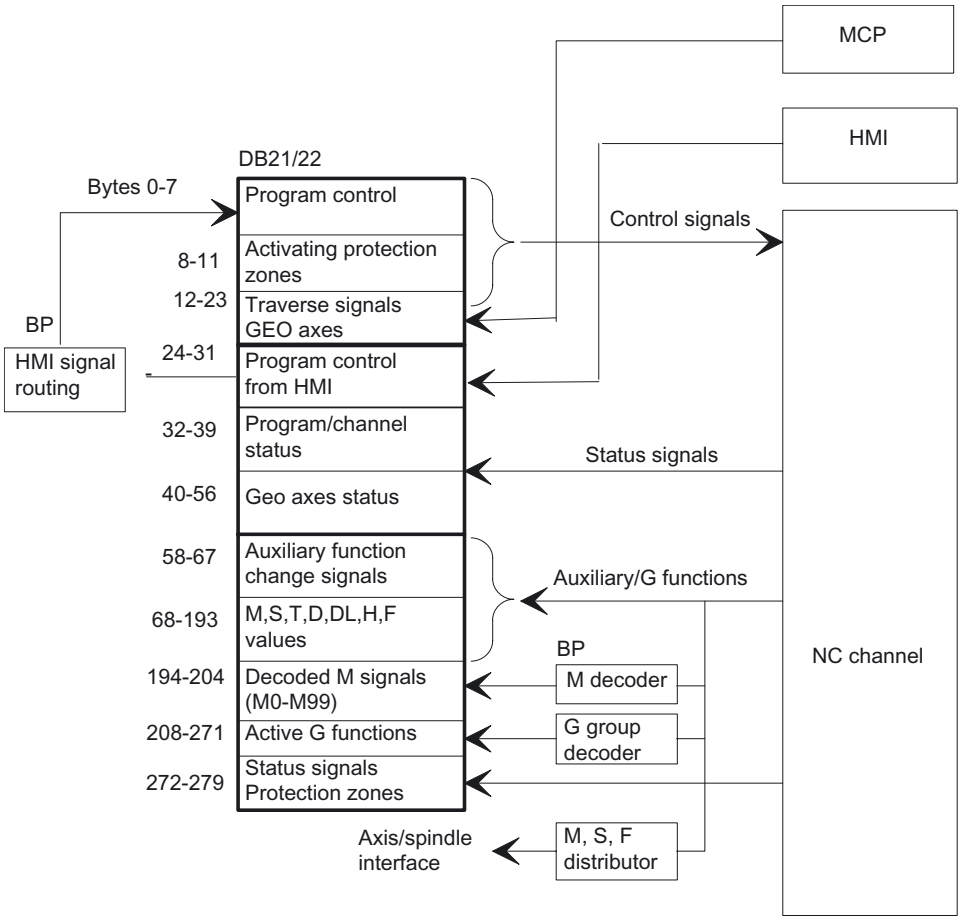


Figure 12-5 PLC/NCK channel interface

PLC/axis, spindle, drive signals

The axis-specific and spindle-specific signals are divided into the following groups:

- Shared axis/spindle signals
- Axis signals
- Spindle signals
- Drive signals

The signals are transmitted cyclically at the start of OB 1 with the following exceptions:

Exceptions include:

- Axial F value
- M value
- S value

An **axial F value** is entered via the M, S, F distributor of the basic program if it is transferred to the PLC during the NC machining process.

The **M and S value** are also entered via the M, S, F distributor of the basic program if one or both values require processing.

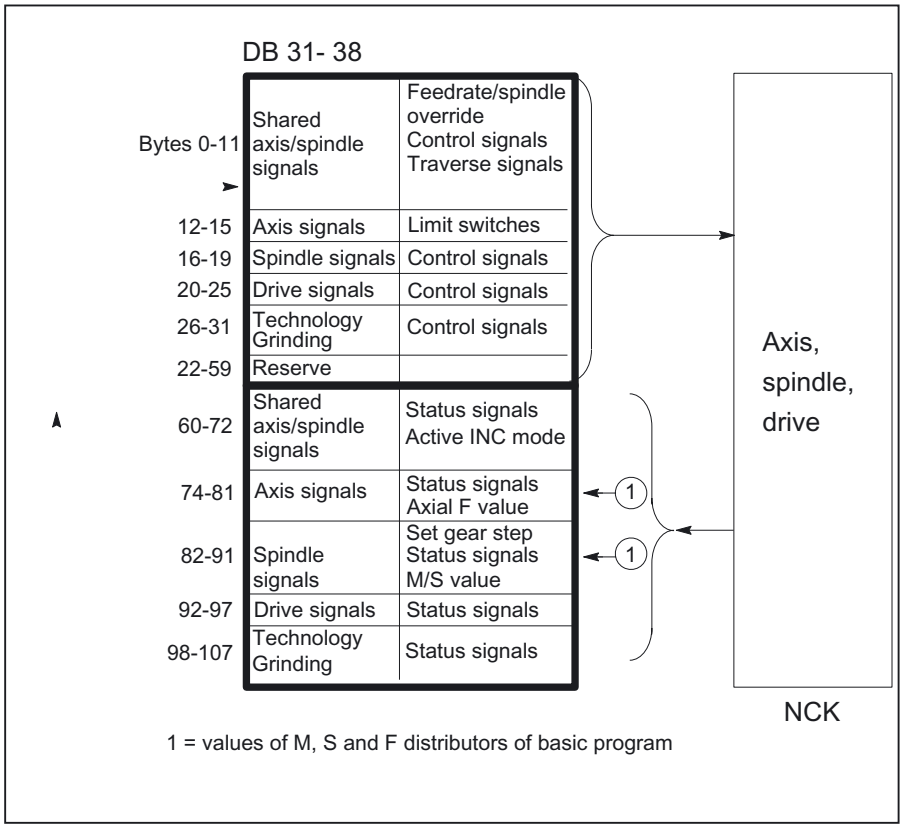


Figure 12-6 Interface between PLC and axes/spindles/drives

12.9.2 Interface PLC/HMI

General

The following groups of functions are required for the PLC/HMI interface:

- Control signals
- Machine operation
- PLC messages
- PLC status display

Control signals

Some control signals are signal inputs, for example, via the machine control panel, which have to be taken into account by the HMI. This group of signals includes, for example, display actual values in MCS or WCS, key disable, etc. These are exchanged with the HMI via a separate interface data block (DB19).

Machine operation

All operator inputs, which lead to response actions on the machine, are monitored by the PLC. Operator actions are usually performed on the machine control panel (MCP). However, it is also possible to perform some operator actions on the HMI, e.g. mode selection.

The PLC operating system enters the operating signals sent by the HMI directly into the interface data blocks. As standard, the basic program routes these operating signals in such a way that, provided equivalent operator actions are available, these can be performed either on the HMI **or** on the MCP. If required, the user can switch off the operation via HMI through a parameter "MMCToIF" of FB1.

PLC messages

The signaling functions are based on the system diagnostic functions integrated in the operating system of the AS 300. These have the following characteristics:

- The PLC operating system enters all important system states and state transitions in a **diagnostics status list**. Communication events and I/O module diagnostics data (for modules with diagnostic functions) are also entered.
- Diagnostics events, which lead to a system stop, are also entered with a time stamp in a **diagnostic buffer** (circular buffer) in the chronological order of their occurrence.
- The events entered in the diagnostic buffer are automatically transmitted to human machine interface systems (OP or HMI) via the bus systems once these have issued a ready signal (message service). Transfer to the node ready is a function of the PLC operating system. Receipt and interpretation of the messages are executed by the HMI software.
- The PLC user program can also use SFCs (System Function Calls) to enter messages in the diagnostic buffer or ALARM S/ALARM SQ buffer.
- The events are entered in the interrupt buffer.

  The associated message texts must be stored on the OP or HMI.

An FC (FC 10) for message acquisition is prepared in conjunction with the basic program. This FC records events, subdivides them into signal groups and reports them to the HMI via the interrupt buffer.

The message acquisition structure is shown in the figure "Acquisition and signaling of PLC events". The features include:

- Bit fields for events related to the NC/PLC interface are combined in a single data block (DB2) with bit fields for user messages.
- Bit fields are evaluated at several levels by FC10.
  - **Evaluation 1;** Acquisition of **group signals**

    A group signal is generated for each group of signals if at least one bit signal is set to "1". This signal is generally linked to the disable signal of the NC/PLC interface (on modules with diagnostic functions). The group signals are acquired completely in cycles.
  - **Evaluation 2;** Acquisition of **interrupt messages**

    A fixed specification exists to define which signals in a group generate an interrupt message when they change from "0" to "1".
  - **Evaluation 3;** Acquisition of **operating signals**

    A fixed specification exists to define which signals in a group generate an operational message.
- The scope of the user bit fields (user area) is set by default to 10 areas with 8 bytes each, but the number of areas can also be adjusted to suit the requirements of the machine manufacturer via basic program parameters in FB 1.

Acknowledgement concept

The following acknowledgement procedures are implemented for error and operational messages:

Operating messages are intended for the display of normal operating states as information for the user. Acknowledgement signals are, therefore, not required for this type of message. An entry is made in the diagnostic status list for incoming and outgoing messages. The HMI maintains an up-to-date log of existing operating messages using the identifiers "operating message arrived" and "operating message gone".

Interrupt messages are used to display error states on the machine, which will usually lead to the machine being stopped. Where several errors occur in rapid succession, it is important to be able to distinguish their order of occurrence for troubleshooting purposes. This is indicated, on the one hand, by the order in which they are entered in the diagnostic buffer and on the other, by the time stamp, which is assigned to every entry.

If the cause of the error disappears, the associated interrupt message is only deleted if the user has acknowledged it (e.g. by pressing a key on the MCP). In response to this signal, the "Message acquisition" FC examines which of the reported errors have disappeared and enters these in the diagnostic buffer with the entry "Interrupt gone". This enables the HMI to also maintain an up-to-date log of pending interrupt messages. The time of day indicating the time at which the error occurred is maintained for messages, which are still pending (in contrast to a received interrogation).

STEP 7

A tool can be started in the SIMATIC Manager via menu item "Target system" > "CPU messages". Alarms and messages can be displayed by number using this tool. To do this, acivate the "Alarm" tab and enter a check mark under "A" in the upper half of the screen.

User program

The user PLC program merely needs to call the basic program block FC 10 with appropriate parameter settings in the cyclic program section and set or reset the bit fields in DB2. All further necessary measures are implemented by the basic program and HMI.

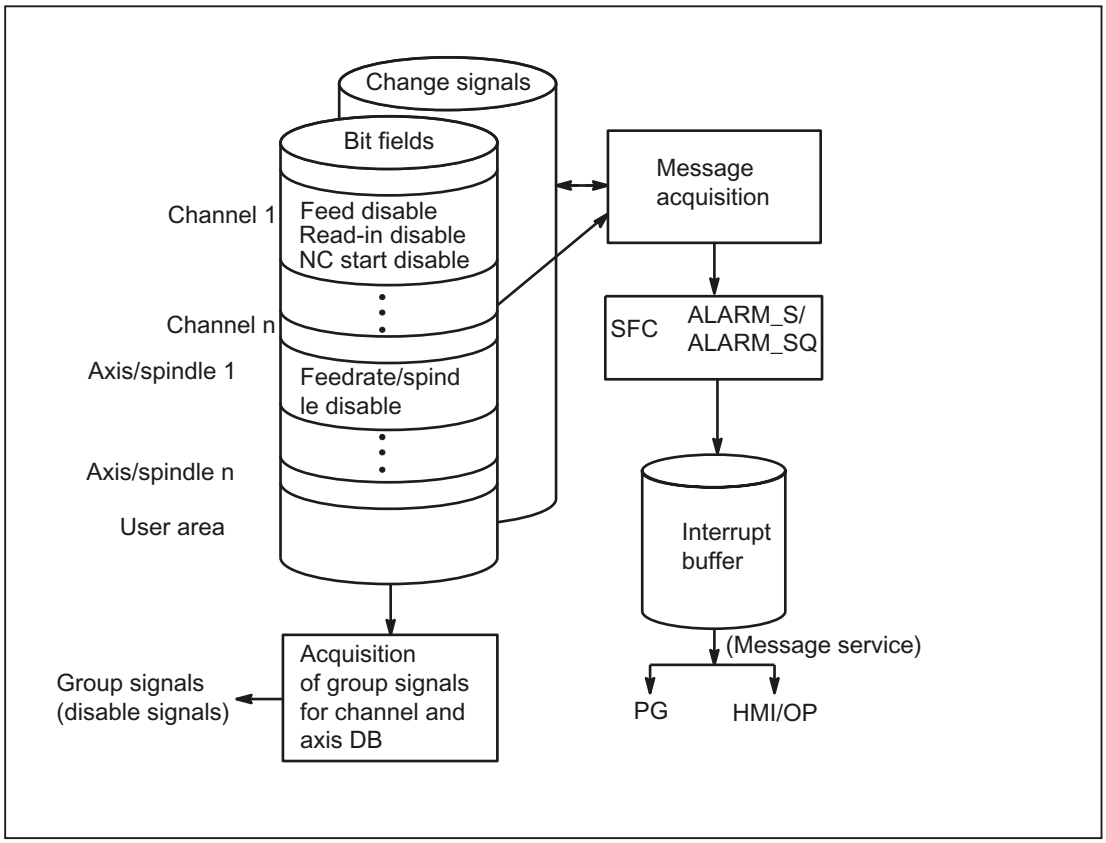


Figure 12-7 Acquisition and signaling of PLC events

Extensions of the PLC alarms via the block FC 10

The FB 1 parameter "ExtendAlMsg" helps in making a selection of the PLC alarm mechanism.

If "ExtendAlMsg:= FALSE" the earlier process of the FC 10 with the DB 2 is active as bit-field data block. The known restrictions regarding the number of channels and axes are applicable.

On the other hand, in case of "ExtendAlMsg:= TRUE" the extension of the FC 10 becomes active. The DB 2 and DB 3 are created as usual. The user must set or reset the bits in DB 2. The parameter setting via message and alarm and a parameter setting of the numeric value of the 2nd decade of the user alarms are contained in DB 5.

The extensions are:

- Support for 10 channels, 31 axes.
- Areas for feed stop, read-in disable, etc. are available without messages. The information from this area is stored on the interface in DB21, DB31 depending upon the FC 10 parameter "ToUserIF" together with the related message bits as group signals. As such, the previous cumbersome handling of the signals is omitted.
- The alarms / messages also get the 16-bit integer additional value (%Z parameter in the alarm text) in addition to the alarm number for the user area 0. The user must write the 16-bit integer values in the DB 2 in the Array variable ZInfo0 parallel to setting an alarm bit. An integer value is available for each bit in the user area 0, see UDT1002 in the basic program.
- The user messages can be parameteized in the second decade of the message number in the numerical range 0 to 9. The display value of the second decade must be written by the user in the DB5 in the array variable UserDek2No. A number can be defined for each user area, see DB 5 in the basic program.

  The value 0 is set by default for second decade.

The structuring of the DB 2 in UDT1002 can be recoginzed (basic program). In case of new alarm functions, the UDT1002 must be assigned symbolically to the DB2.

At the start of DB 2 there are bit fields for signals without alarm generation. This is followed by an array of size 64 integer for additional info of the user area 0.

Thereafter follow the areas, which also trigger alams / messages (see List manual) These areas are extended to 10 channels, 31 axes.

Simple implementation of a user program on the new alarms

The source container of the basic program contains the file "udt2_for_Convert.awl", which has the following structural element from UDT1002:

- ChanA as array of 1 ... 8
- AxisA as array of 1 ... 18
- UserA as array of 1 ... 31

This UDT2 is to be compiled via the KOP/FUP/AWL - Editor. The UDT2 must be assigned to the DB 2 in the symbol table.

Sources must be generated for components, which have assignments on DB 2. Alternatively, sources can naturally be created for all blocks too. The UDT1002 must now be assigned to the DB 2 in the symbol table. Thereafter, the sources must be recompiled.

Now all the alarm allocations are assigned to the new data areas in the DB 2 and now only the parameter "ExtendAlMsg" at FB 1 must be set to True .

After a Power On RESET the alarm behavior is the same as earlier.

12.9.3 PLC/MCP/HHU interface

General

There are different connection options for the machine control panel (MCP) and the handheld unit (HHU). This is in part due to the history of the MCP and HHU. This description focuses primarily on the connection of the Ethernet components.

On the SINUMERIK 840D sl, the machine control panel (MCP) and handheld unit (HHU) are connected via the Ethernet bus, which also links the TCU to the NCU. The advantage of this is that only one bus cable is required to connect the operating unit.

Topology SINUMERIK 840D sl

On the 840 D, the machine control panel and the handheld unit are connected to the CP 840D sl Ethernet bus (see Figure below). Where the connection of further keys and displays is required for customized operator panels, an additional keyboard interface (machine control panel without operating unit) can be used. For each keyboard interface, 64 pushbuttons, switches, etc. and 64 display elements can be connected via ribbon cable.

The signals sent from the MCP are copied to the PLC's DPR (Dual-Port RAM) by the integrated Ethernet CP-840D sl. The basic program of the PLC enters the incoming signals in the input image configured on FB1. The NC-related signals are generally distributed by the basic program to the NC/PLC interface. If required, the signals can be modified by the user.

The signals from the PLC to the MCP (displays) are transferred in the opposite direction.

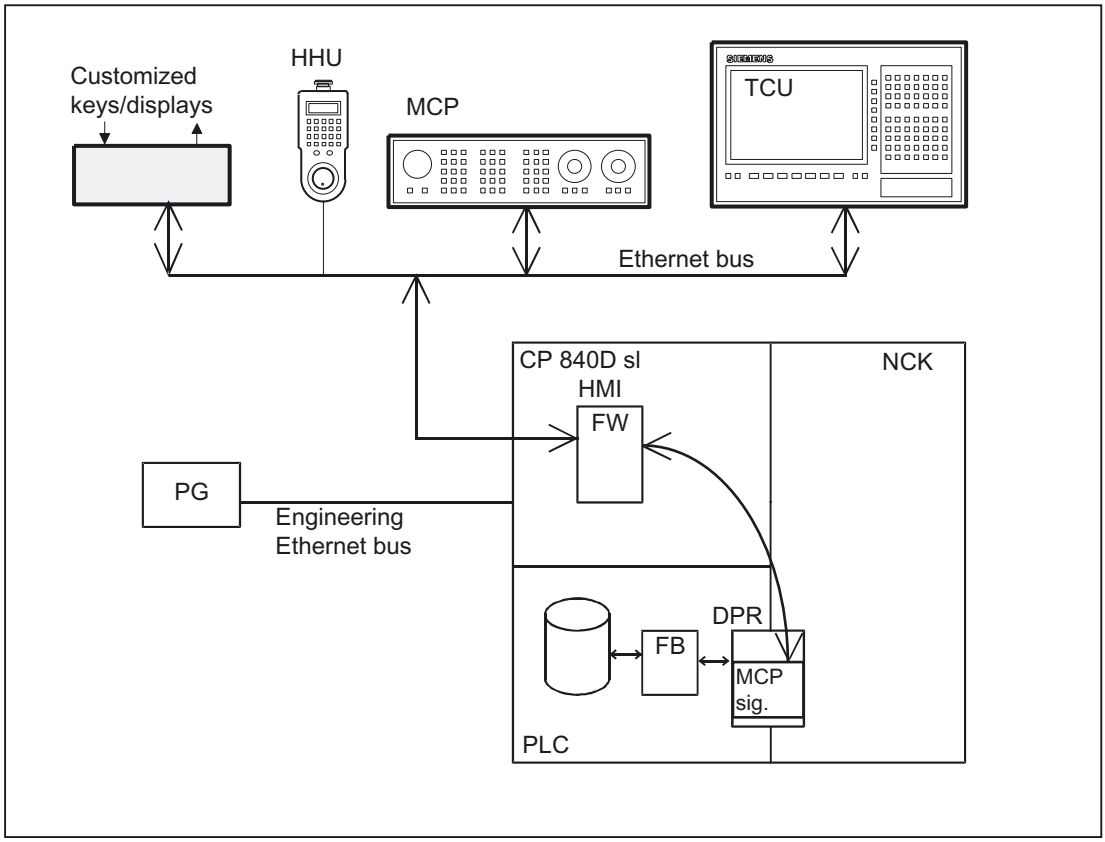


Figure 12-8 Connection of the machine control panel on 840D sl

Bus addresses

On Ethernet components, MAC and IP addresses or logic names are determining factors in respect of communication. The control system's system programs convert logic names into MAC or IP addresses. On the PLC, the numeric component of the logic name is used for communication. This numeric part is specified by the user to the FB 1 via the parameter "MCPxBusAdr".

The logical name of an MCP or HHU always begins with "DIP". This is followed by a number corresponding to the switch position of the MCP component (e.g. DIP 192, DIP 17).

MCP interface in the PLC

The signals from the machine control panel are routed by default via the I/O interface to the PLC area. A distinction must be made between NC and machine-specific signals. NC-specific key signals are distributed to the relevant mode-group-, NCK-, axis- and spindle-specific interface by FC19 (or FC24, FC25, FC26, depending on the type of MCP) by default. The reverse applies to the associated status signals which are routed to the MCP interface. For this purpose, FC 19 or the other blocks mentioned above must be called in the **user program**.

Customized keys, which can be used to trigger a wide range of machine functions, must be evaluated directly by the user program. The user program also routes the status signals to the output area for the LEDs.

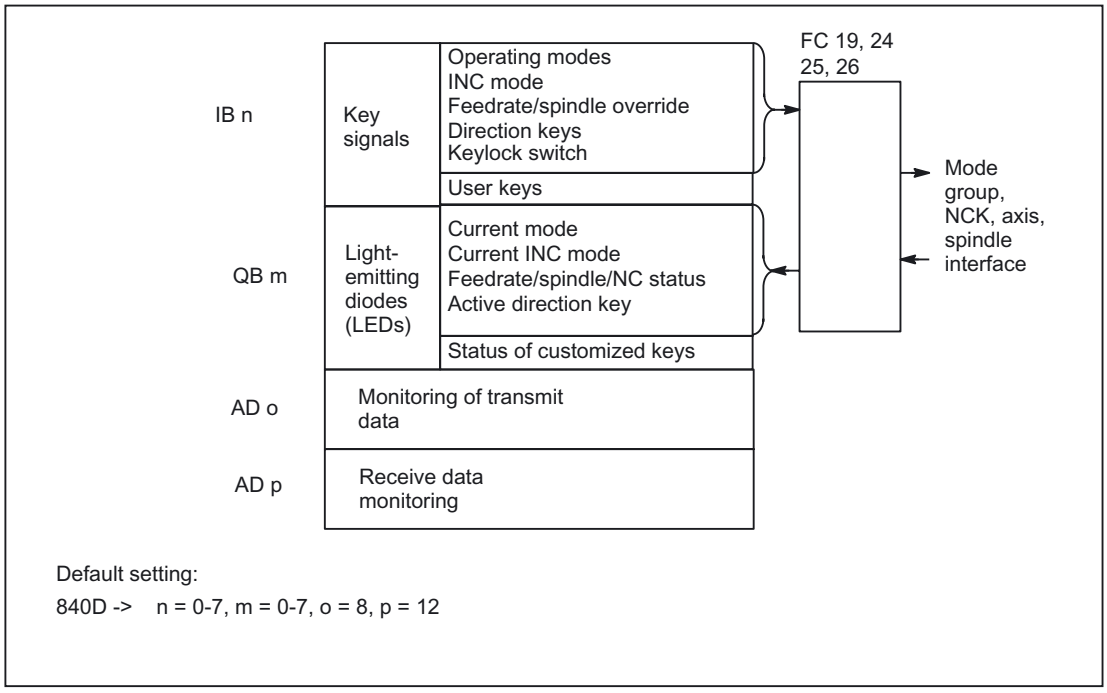


Figure 12-9 Interface to and from machine control panel

12.10 Structure and functions of the basic program

General

The PLC program has a modular structure. The organization blocks (OB) form the interface between the operating system and the basic and user programs.

- Restart (warm restart) with start-up and synchronization (OB 100)
- Cyclic operation (OB 1)
- Process alarms (OB 40)
- Asynchronous errors: Diagnostic alarm (OB 82), module failure (OB 86)

The calls of the function blocks of the basic and user programs must be programmed by the user in the organization blocks (OB).

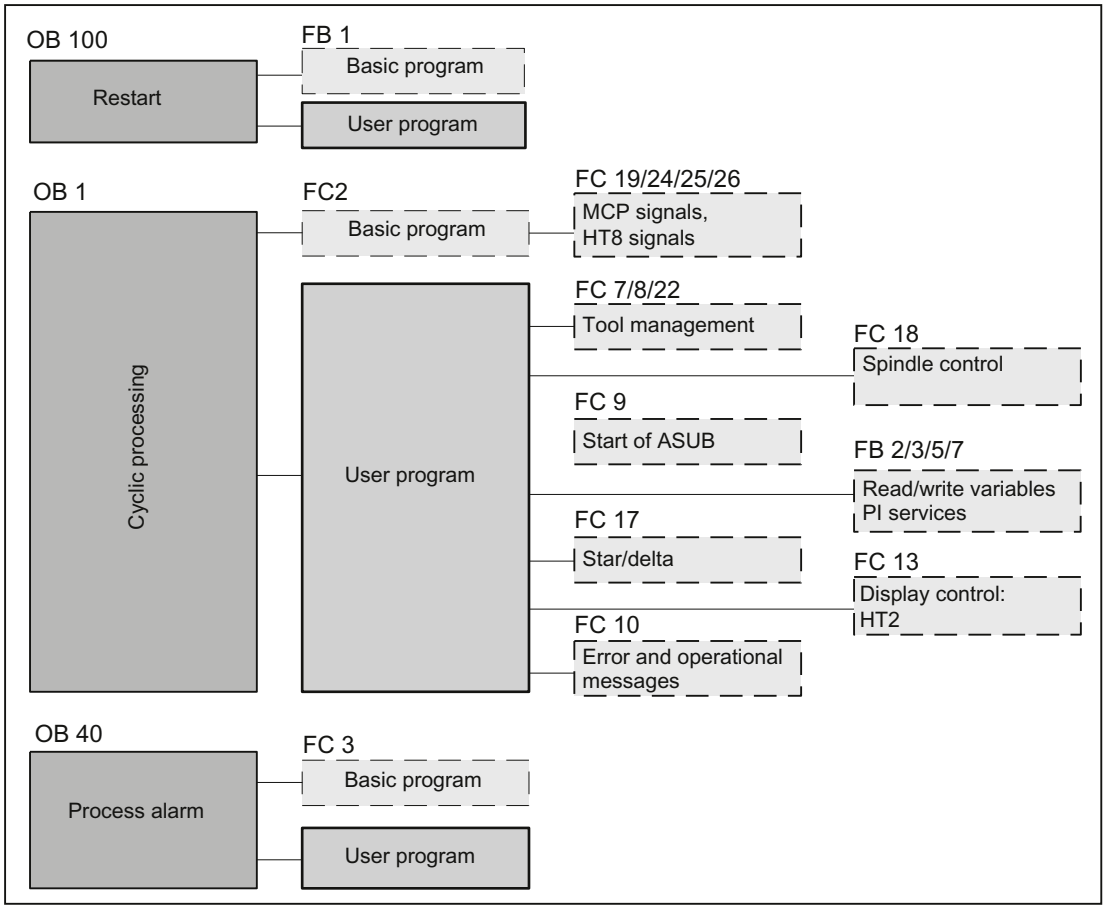


Figure 12-10 Structure of the basic program (principle)

12.10.1 Startup and synchronization of NCK PLC

Loading the basic program

The basic program must be loaded with the S7 tool when the PLC is in the Stop state. This ensures that all blocks in the basic program will be initiated correctly the next time they are called. Otherwise, undefined states can occur in the PLC (e.g. blinking of all PLC LEDs).

Startup,

The synchronization of NCK and PLC is performed during startup. The system and user data blocks are checked for integrity and the most important basic program parameters are verified for plausibility. In cases of errors, the basic program produces an alarm (visible on HMI) and switches the PLC to the Stop state.

A warm restart is not provided, i.e. following system initialization, the operating system runs organization block OB 100 and always commences cyclic execution at the start of OB 1.

Synchronization

The PLC is synchronized with the HMI and NCK and CP during powerup.

Sign-of-life

After a correct initial start and the first complete OB1 cycle (initial setting cycle) the PLC and NCK continuously exchange sign of life signals. If the sign of life from the NCK fails to materialize, the PLC/NCK interface is initialized and the signal "NCK CPU ready" in DB 10 is set to FALSE.

12.10.2 Cyclic operation (OB 1)

General

The NCK PLC interface is processed completely in cyclic mode. From a chronological viewpoint, the basic program runs ahead of the user program. In order to minimize the execution time of the basic program, only the control/status signals are transmitted cyclically; transfer of the auxiliary functions and G functions only takes place on request.

The following functions are performed in the cyclic part of the basic program:

- Transmission of the control/status signals
- Distribution of the auxiliary functions
- M decoding (M00 - M99),
- M, S, F distribution
- Transfer the MCP signals via NCK
- Acquisition and conditioning of the user errors and operating messages.

Control/status signals

A shared feature of the control and status signals is that they are bit fields. The basic program updates them at the start of OB1.

The signals can be subdivided into the following groups:

- General signals
- Mode group-specific signals (such as mode types)
- Channel-specific signals (such as program and feed modifications)
- Axis- and spindle-specific signals (such as feed disable)

Auxiliary and G functions

The auxiliary and G functions have the following characteristics:

- Transfer to the PLC is block-synchronous (referred to a part program block)
- Transfer is acknowledge-driven.
- The acknowledgement times have an immediate effect on the execution time of NC blocks containing auxiliary functions requiring acknowledgement.

The value range is presented in the table below:

Function	Structure		Range of values		Data type	
	1st value	2nd value	1st value	2nd value	1st value	2nd value
G function		G function		255[1)]		Byte
M word	M group	M word	99	99,999,999	Word	DWord
S word	Spindle no.	S word	6	Floating point[2)]	Word	DWord
T word	Magazine no.	T word	99	65535	Word	Word
D word	-	D word	99	255	Byte	Byte
H word	H group	H word	99	Floating point	Word	DWord
F word	Axis no.	F word	18	Floating point	Word	DWord

1) relative number, transferred for each G group

2) corresponding STEP 7 format (24-bit mantissa, 8-bit exponent)

The M, S, T, H, D and F values sent by the NCK are output together with the accompanying change signals to the **CHANNEL DB** interface via the auxiliary/G functions (see List Manual). The function value and the extended address are transferred to the appropriate data word. The accompanying modification signal is activated to 1 for one PLC cycle. When the modification signal is reset, the acknowledgement is passed to the NCK. The acknowledgement of high-speed auxiliary functions is given by the basic program immediately the basic program detects the auxiliary function.

In addition to distribution of the auxiliary and G functions, selected signals are processed as described below.

M decoder

M functions can be used to transfer switching commands and fixed-point values. Decoded dynamic signals are output to the **CHANNEL DB** interface for standard M functions (range M00 - M99) signal length = 1 cycle time).

G group decoders

In the case of G functions sent by the NCK, the related groups are decoded and the current G number is entered in the corresponding interface byte of the CHANNEL DB, i.e. all active G functions are entered in the channel DBs. The entered G functions are retained even after the NC program has terminated or aborted.

Note

During system startup, all G group bytes are initialized with the value "0".

M, S, F distributor

The M, S, F, distributor is used to enter spindle-specific M words M(1...6)=[3,4,5], S words and F words for axial feeds in the appropriate **spindle and axis data blocks**. The criterion for distribution is the extended address which is passed to the PLC for M words, S words and axial F words.

MCP signal transmission

Depending on the bus connection, the MCP signals are either transmitted directly to the PLC or indirectly to the parameterized I/O areas via an internal procedure using the basic program.

User messages

The acquisition and processing of the user error and operational messages is performed by an FC in the basic program.

12.10.3 Time-interrupt processing (OB 35)

The user must program **OB 35** for time-alarm processing. The default time base setting of OB 35 is 100 ms. A different time base can be selected using the STEP7 "HW Config" tools. However, the OB 35 time setting must be at least 3 ms in order to avoid a PLC CPU stop. The stop is caused by reading of the HMI system state list during powerup of the HMI. This reading process blocks priority class control for approx. 2 ms. The OB 35 with a time base set to a rather lower value is then no longer processed correctly.

12.10.4 Process interrupt processing (OB 40)

A process interrupt **OB 40** (interrupt) can, for example, be triggered by appropriately configured I/Os or by certain NC functions. Due to the different origin of the interrupt, the PLC user program must first interpret the cause of the interrupt in OB 40. The cause of the interrupt is contained in the local data of OB 40.

References:
SIMATIC STEP 7 Description or Online Help of STEP 7

12.10.5 Diagnostic interrupt, module failure processing (OB 82, OB 86)

General

A module diagnosis or module failure on an I/O module triggers OB 82 / OB 86. These blocks are supplied by the basic program. The basic program block FC5 is called in these OBs. This is wired by default to trigger a PLC stop in the event of an error being detected.

A PLC stop:

- If the cause of the error is removed, a PLC stop will not be triggered.
- does not occur in case of the PROFIBUS-MCPs specified as the FB 1 parameters.

The response can be changed by modifying the FC5 parameter setting.

PROFIBUS diagnostics

The slaves of both the PROFIBUS connections MPI/DP or DP1 are registered by the basic program as group message in the form of a ready signal in the interface signal

DB10 DBX92.0 (MPI/DP Bus Slaves OK) and

DB10 DBX92.1 (DP1 Bus Slaves OK)

The group message is derived from the LED status of the respective PROFIBUS (System state list SZL 0x174).

If an error or a failure of a slave is detected, the alarm 400551 or 400552 is also reported. The alarm is automatically deleted when the error is removed. Detailed information about the error or the failure will be displayed in the alarm messages as exact cause of error in a future HMI software version in the diagnostics. No FC 125, FB 125 is necessary for this diagnostic display, because the information is called directly from the HMI via the corresponding SZL functions of the PLC operating system via the communication channels to the PLC. This saves the relatively large block in the PLC and reduces also the PLC cycle time. Since the FB 125 works under interrupt disable during its processing, OBs with a higher priority can be executed more quickly.

Note

Until the HMI software version with the diagnostic display for the PROFIBUS becomes available, it is recommended to use FC 125 for PROFIBUS diagnostics. This block has relatively less cycle time and the slave states can be updated via the DB 10 interface signal mentioned earlier by activating the FC 125 block.

12.10.6 Response to NCK failure

General

During cyclic operation, the PLC basic program continuously monitors NCK availability by polling the signoflife character. If the NCK is no longer reacting, the NCK PLC interface is initialized, and the **NCK CPU ready** signal in the area of the **signals from NC (DB 10.DBX 104.7)** is reset. Furthermore, the signals sent from the NCK to the PLC and vice versa are set to an initial state.

The PLC itself remains active so that it can continue to control machine functions. However, it remains the responsibility of the user program to set the machine to a safe state.

NCK → PLC signals

The signals sent by the NCK to the PLC are divided into the following groups:

- Status signals from the NCK, channels, axes and spindles
- Modification signals of the auxiliary functions
- Values of the auxiliary functions
- Values of the G functions

Status signals:

The status signals from the NCK, channels, axes, and spindles are reset.

Auxiliary-function modification signals:

Auxiliary-function modification signals are also reset.

Auxiliary-function values:

Auxiliary-function values are retained so that it is possible to trace the last functions triggered by the NCK.

G-function values:

G function values are reset (i.e. initialized with the value 0 respectively).

PLC → NCK signals

The signals sent by the PLC to the NCK are divided into control signals and tasks that are transferred by FCs to the NCK.

Control signals:

The control signals from the PLC to the NCK are frozen; cyclic updating by the basic program is suspended.

Jobs from PLC to NCK:

The FCs and FBs, which are used to pass jobs to the NCK, must no longer be processed by the PLC user program, as this could lead to incorrect checkback signals. During powerup of the control, a job (e.g. read NCK data) must not be activated in the user program until the **NCK CPU ready** signal is set.

12.10.7 Functions of the basic program called from the user program

General

In addition to the modules of the basic program which are called at the start of OBs 1, 40 and 100, functions are also provided which have to be called and supplied with parameters at a suitable point in the user program.

These functions can be used, for example, to pass the following jobs from the PLC to the NCK:

- Traversing concurrent axes (FC 18)
- Start asynchronous subprograms (ASUBs) (FC 9),
- Selecting NC programs (FB 4)
- Control of spindles (FC 18),
- Read/write variables (FB 2, FB 3)

Note

Checking and diagnostics of a function call of the basic PLC program

To simplify the checking and diagnostics of a function call (FB or FC) of the basic PLC program that is controlled via a trigger (e.g. via Req, Start parameters) and that provide an execution acknowledgement as output parameter (e.g. via Done, NDR, Error parameters), proceed as follows.

A variable compiled of other signals which produce the trigger for the function call should be set. Start conditions may be reset only as a function of the states of parameters Done, NDR and Error.

The appropriate control mechanism can be placed in front of or behind the function call. If the mechanism is placed after the call, the output variables can be defined as local variables (advantage: Reduction of global variables, markers, data variables and time-related advantages over data variables).

The trigger parameter must be a global variable (e.g. marker, data variable).

Jobs that are still active must be reset from the user program in OB 100 (Req, Start, parameters,
etc. from TRUE ⇒ FALSE). A POWER OFF/ON could result in a state in which jobs are still active.

Concurrent axes

The distinguishing features of concurrent axes are as follows:

- They must be defined as such via the NC machine data.
- They can be traversed either from the PLC or from the NC by means of the JOG keys.
- Starting from the PLC is possible in the NC operating modes MDA and AUTOMATIK via FC.
- The start is independent of NC block boundaries.

Function calls are available for positioning axes, indexing axes and spindles (FC 18).

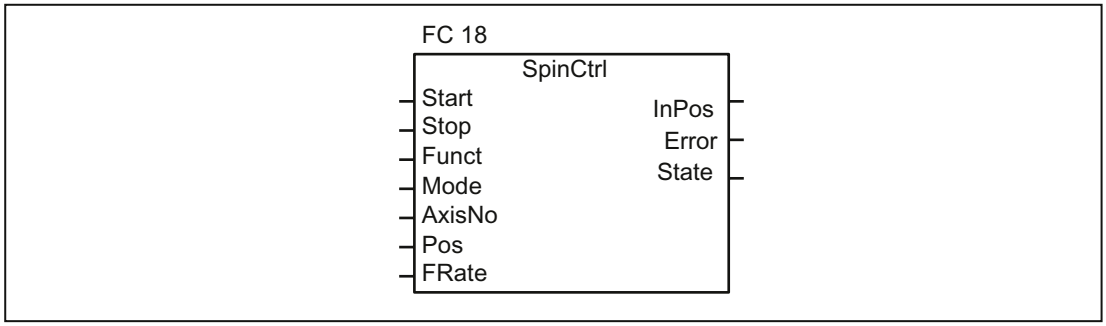
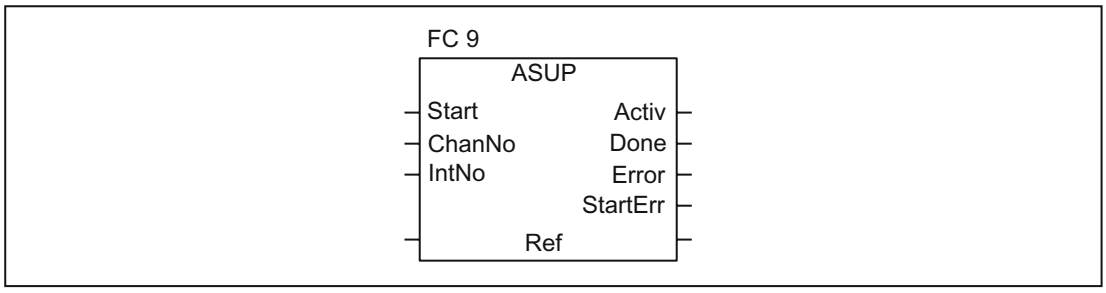


Figure 12-11 FC 18 input/output parameters

Asynchronous subprograms (ASUBs)

An ASUB can be used to activate arbitrary functions in the NCK. Before an asynchronous subprogram can be started from the PLC, it must be ensured that it is available and prepared by the NC program or by FB 4 PI services (ASUB).

Once prepared in this way, it can be started at any time from the PLC. The NC program running in one of the parameterized channels of FC 9 is interrupted by the asynchronous subprogram. An ASUB is started by calling FC 9 from the user program by setting the start parameter to 1.



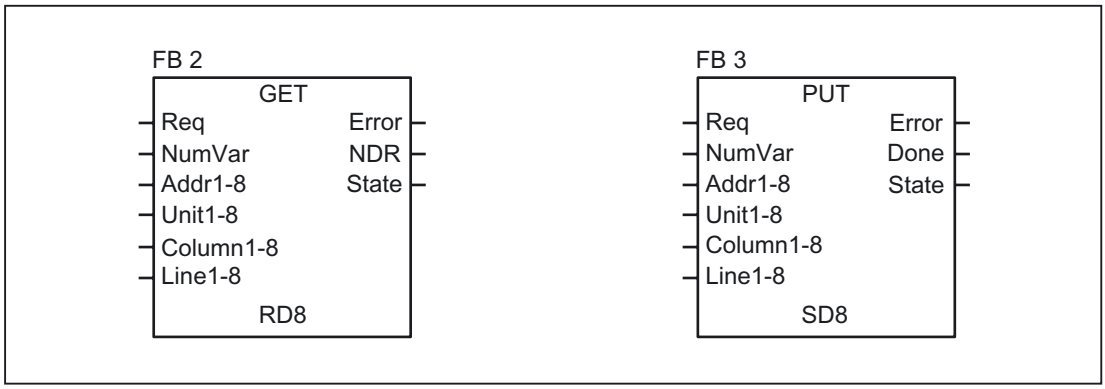
Note

If an asynchronous subprogram has not been prepared by an NC program or by FB 4 (ASUB) (e.g. if no interrupt no. has been assigned), a start error is output (StartErr = TRUE).

Read/Write NC variables

NCK variables can be read with FB GET while values can be entered in NCK variables with FB PUT. The NCK variables are addressed via identifiers at inputs Addr1 to Addr8. The identifiers (symbols) point to address data which must be stored in a global DB. To allow generation of this DB, the PC software is supplied with the basic program with which the required variables can be selected from a table, which is also supplied. The selected variables are first collected in a second, project-related list. Command **Generate DB** creates a "*.AWL" file which must be linked to the program file for the machine concerned and compiled together with the machine program.

1 to 8 values can be read or written with a read or write job. If necessary, the values are converted [e.g. NCK floating-point values (64-bit) are converted to PLC format (32-bit with 24-bit mantissa and 8-bit exponent) and vice versa]. A loss of accuracy results from the conversion from 64-bit to 32-bit REAL. The maximum precision of 32-bit REAL numbers is approximately 10 to the power of 7.



AG_SEND/AG_RECV functions

The AG_SEND/AG_RECV functions correspond to the functions of the library "SIMATIC_NET_CP" of the S7-300 CPU in STEP 7. In general, the online help is valid for these functions.
The AG_SEND/AG_RECV functions can be used for data exchange with another station via the integrated "CP 840D sl". A description of the functions is provided in Section "Block descriptions (Page 999)".

Note

Other communication blocks (e.g. BSEND, USEND) which possess a CP343-1 are not supported in SINUMERIK 840D sl.

12.10.8 Symbolic programming of user program with interface DB

General

Note

The basic program library on the CD supplied with the Toolbox for the 840D contains files NST_UDTB.AWL and TM_UDTB.AWL.

The compiled UDT blocks from these two files are stored in the CPU program of the basic program.

A UDT is a data type defined by the user that can, for example, be assigned to a data block generated in the CPU.

Symbolic names of virtually all the interface signals are defined in these UDT blocks.

The UDT numbers 2, 10, 11, 19, 21, 31, 71, 72, 73 are used.

The assignments have been made as follows:

UDT assignments		
UDT number	**Assignment to interface DB**	**Significance**
UDT 2	DB 2	Interrupts/Messages
UDT 10	DB 10	NCK signals
UDT 11	DB 11	Mode group signals
UDT 19	DB 19	HMI signals
UDT 21	DB 21 to DB 30	Channel signal
UDT 31	DB 31 to DB 61	Axis/spindle signals
UDT 71	DB71	Tool management: Load/unload locations
UDT 72	DB 72	Tool management: Change in spindle
UDT 73	DB 73	Tool management: Change in revolver
UDT 77	DB 77	MCP and HHU signals with standard SDB 210
UDT 1002	DB 2	extended alarms / messages (FB 1-Parameter "ExtendAlMsg:=TRUE"

To symbolically program the interface signals, the interface data blocks must first be symbolically assigned using the symbol editor.

For example, symbol "AxisX" is assigned to operand DB31 with data type UDT 31 in the symbol file.

After this input, the STEP 7 program can be programmed in symbols for this interface.

Note

Programs generated with an earlier software version that utilize the interface DBs described above can also be converted into symbol programs. A fully qualified instruction for data access e.g. "U DB31.DBX60.0" is then necessary here for (spindle/ no axis) in the program created till now. This command is converted upon activation of the symbolics in the editor "AxisX.E_SpKA".

Description

Abbreviated symbolic names of the interface signals are defined in the two STL files NST_UDTB.AWL and TM_UDTB.AWL.

In order to create the reference to the names of the interface signals, the name is included in the comment after each signal.

The names are based on the English language. The comments are in English.

The symbolic names, commands and absolute addresses can be viewed by means of a STEP 7 editor command when the UDT block is opened.

Note

Unused bits and bytes are listed, for example, with the designation "f56_3".

- "56": Byte address of the relevant data block
- "3": Bit number in this byte

12.10.9 M decoding acc. to list

Description of functions

When the **M decoding according to list** function is activated via the GP parameter of FB1 "ListMDecGrp" (number of M groups for decoding), up to 256 M functions with extended address can be decoded by the basic program.

The assignment of the M function with extended address and the bit to be set in the signal list is defined in the decoding list. The signals are grouped for this purpose.

The signal list contains 16 groups with 16 bits each as decoded signals.

There is only one decoding list and one signal list i.e. this is a cross-channel function.

The M functions are decoded. Once they are entered in the decoding list, then the associated bit in the signal list is set.

When the bit is set in the signal list, the readin disable in the associated NCK channel is set simultaneously by the basic program.

The readin disable in the channel is reset once the user has reset all the bits output by this channel and thus acknowledged them.

The output of an M function decoded in the list as a highspeed auxiliary function does not result in a readin disable.

The figure below shows the structure of the **M decoding according to list**:

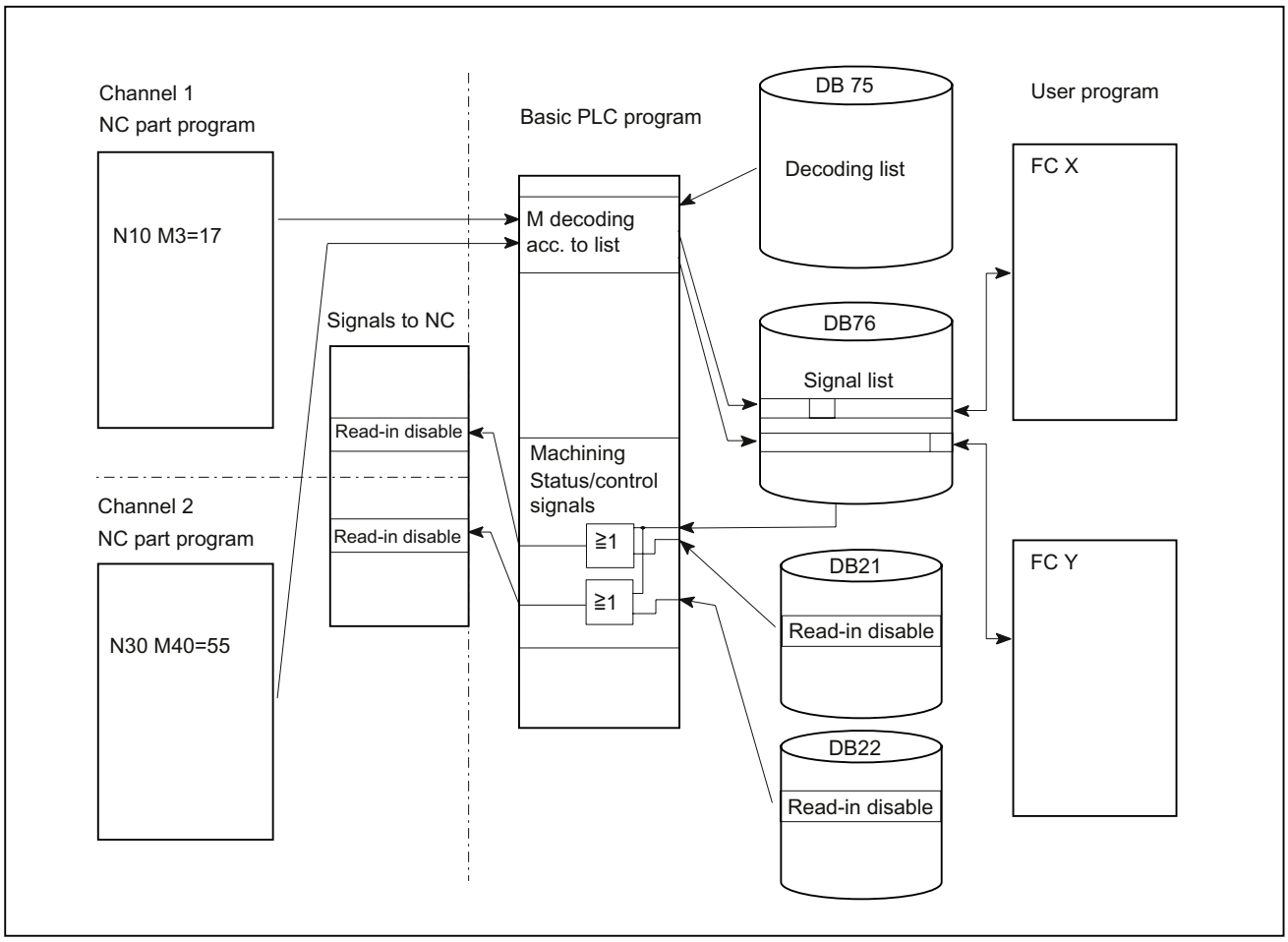


Figure 12-12 M decoding acc. to list

Activation of the function

The number of groups to be evaluated / decoded is indicated in the basic program parameter "ListMDecGrp" when FB 1 is called in OB 100 (see also " FB 1: RUN_UP Basic program, startup section (Page 999) "). M decoding is activated if this value is between 1 and 16. Before the function is activated, the decoding list DB75 must be transferred to the PLC followed by a restart.

Structure of decoding list

The source file for the decoding list (MDECLIST.AWL) is supplied with the basic program. DB 75 is created when the STL source is compiled.

There must be an entry in decoding list DB 75 for every group of M functions to be decoded.

A maximum of 16 groups can be created.

16 bits are available in each group in the list of decoded signals.

The assignment between the M function with extended address and the bit to be set in the signal list is specified via the first and last M functions in the decoding list.

The bit address is generated correspondingly from the first M function ("MFirstAdr") to the last M function ("MLastAdr") from bit 0 up to maximum bit 15 for each group.

Each entry in the decoding lists consists of 3 parameters, each of which is assigned to a group.

Assignment of groups			
Group	**Extended M address**	**First M address in group**	**Last M address in group**
1	MSigGrp[1].MExtAdr	MSigGrp[1].MFirstAdr	MSigGrp[1].MLastAdr
2	MSigGrp[2].MExtAdr	MSigGrp[2].MFirstAdr	MSigGrp[2].MLastAdr
...	...	...	...
16	MSigGrp[16].MExtAdr	MSigGrp[16].MFirstAdr	MSigGrp[16].MLastAdr

Type and value range for signals			
Signal	**Type**	**Value range**	**Remark**
MExtAdr	Int	0 ... 99	Extended M address
MFirstAdr	DInt	0 to 99.999.999	First M address in group
MLastAdr	Dint	0 to 99.999.999	Last M address in group

Signal list

Data block DB 76 is set up when the function is activated.

A bit is set in the appropriate group in DB 76 for an M signal decoded in the list.

At the same time, a readin disable is set in the channel in which the M function has been output.

Example

Three groups of M commands are to be decoded in the following example:

- M2 = 1 to M2 = 5
- M3 = 12 to M3 = 23
- M40 = 55

Structure of the decoding list in DB 75:

Example parameters				
Group	**Decoding list (DB 75)**			**Signal list**
	Extended M address	First M address in group	Last M address in group	DB 76
1	2	1	5	DBX 0.0 ... DBX 0.4
2	3	12	23	DBX 2.0 ... DBX 3.3
3	40	55	55	DBX 4.0

```
DATA_BLOCK DB 75
TITLE =
VERSION : 0.0
      STRUCT
            MSigGrp : ARRAY [1 .. 16 ] OF STRUCT
                  MExtAdr :       INT ;
                  MFirstAdr :     DINT;
                  MLastAdr :      DINT;
            END_STRUCT;
      END_STRUCT;
BEGIN
      MSigGrp[1].MExtAdr :=         2;
      MSigGrp[1].MFirstAdr :=      L#1;
      MSigGrp[1].MLastAdr :=       L#5;
      MSigGrp[2].MExtAdr :=         3;
      MSigGrp[2].MFirstAdr :=      L#12;
      MSigGrp[2].MLastAdr :=       L#23;
      MSigGrp[3].MExtAdr :=         40;
      MSigGrp[3].MFirstAdr :=      L#55;
      MSigGrp[3].MLastAdr :=       L#55;
END_DATA_BLOCK
```

Structure of FB 1 in the OB 100

(enter the number of M groups to be decoded in order to activate the function):

```
Call FB 1, DB 7(
...
ListMDecGrp := 3,        //M decoding of three groups
...
);
```

The appending of the entry in OB 100 and transfer of DB 75 (decoding list) to the AG must be followed by a restart. During the restart, the basic program sets up DB76 (signal list).

If the NC program is started at this point and the expanded M function (e.g. `M3=17`) is processed by the NCK, this M function will be decoded and bit 2.5 set in DB 76 (see decoding list DB 75). At the same time, the basic program sets the read-in disable and the processing of the NC program is halted (in the corresponding NC-channel DB the entry "expanded address M function" and "M function no." is made).

The readin disable in the channel is reset once the user has reset and, therefore, acknowledged, all the bits output by this channel in the signal list (DB 76).

12.10.10 PLC machine data

General

The user has the option of storing PLC-specific machine data in the NCK. The user can then process these machine data after the power-up of the PLC (OB 100). This enables, for example, user options, machine expansion levels, machine configurations, etc., to be implemented.

The interface for reading these data lies in the DB 20. However, DB20 is set up by the basic program during power-up only when user machine data are used i.e. sum of GP parameters "UDInt", "UDHex" and "UDReal" is greater than zero.

The sizes of the individual areas, and hence the total length of the DB 20, is set by the following PLC machine data:

MD14504 $MN_MAXNUM_USER_DATA_INT

MD14506 $MN_MAXNUM_USER_DATA_HEX

MD14508 $MN_MAXNUM_USER_DATA_FLOAT

These settings are specified to the user in the GP parameters "UDInt", "UDHex" and "UDReal".

The data is stored in the DB 20 by the BP in the sequence:

1. Integer MD
2. Hexa-fields MD
3. Real MD

The integer and real values are stored in DB 20 in S7 format.

Hexadecimal data is stored in DB20 in the order in which they are input (use as bit fields).

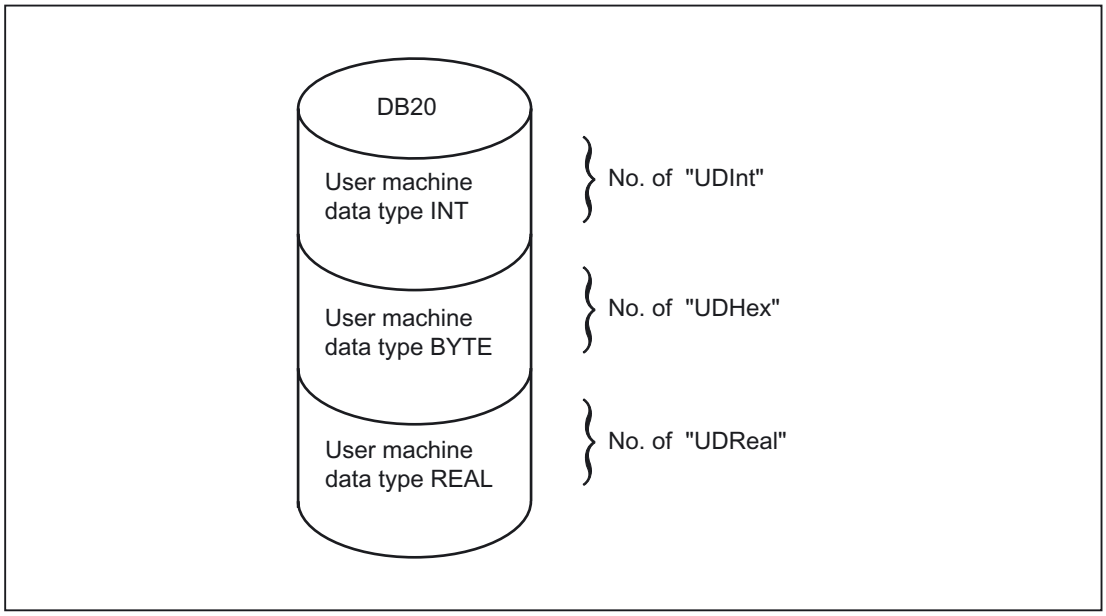


Figure 12-13 DB 20

Note

If the number of PLC machine data used is increased later, then DB20 must be deleted beforehand. To prevent such extensions in use having any effect on the existing user program, the data in DB20 should be accessed in symbolic form wherever possible, e.g. by means of a structure definition in the UDT.

Alarms	
400120	**Delete DB 20 in PLC and restart**
Explanation	DB length is not the same as the required DB length
Reaction	Alarm display and PLC stop
Remedy	Delete DB 20 followed by RESET
Continuation	After cold restart

Example

The project in the example requires 4 integer values, 2 hexadecimal fields with bit information and 1 real value.

Machine data:

MD14510 $MN_USER_DATA_INT[0]	123
MD14510 $MN_USER_DATA_INT[1]	456
MD14510 $MN_USER_DATA_INT[2]	789
MD14510 $MN_USER_DATA_INT[3]	1011
...	
MD14512 $MN_USER_DATA_HEX[0]	12
MD14512 $MN_USER_DATA_HEX[1]	AC
...	
MD14514 $MN_USER_DATA_FLOAT[0]	123.456

GP parameter (OB 100):

```
CALL FB 1, DB 7 (
             MCPNum :=                  1,
             MCP1In :=                  P#E0.0,
             MCP1Out :=                 P#A0.0,
             MCP1StatSend :=            P#A8.0,
             MCP1StatRec :=             P#A12.0,
             MCP1BusAdr :=              6,
             MCP1Timeout :=             S5T#700MS,
             MCP1Cycl :=                S5T#200MS,
             NCCyclTimeout :=           S5T#200MS,
             NCRunupTimeout :=          S5T#50S;
```

BP parameters (to scan runtime):

```
            l gp_par.UDInt;             //=4,
            l gp_par.UDHex;             //=2,
            l gp_par.UDReal;            //=1 )
```

During PLC power-up, DB20 was generated with a length of 28 bytes:

DB 20	
Address	**Data**
0.0	123
2.0	456
4.0	789
6.0	1011
8.0	b#16#12
9.0	b#16#AC
10.0	1.234560e+02

The structure of the machine data used is specified in a UDT:

```
TYPE UDT 20
   STRUCT
       UDInt :     ARRAY [0 .. 3] OF INT;
       UDHex0 :    ARRAY [0 .. 15]OF BOOL;
       UDReal :    ARRAY [0 .. 0] OF REAL;        //Description as field, for
                                                  // later expansions
   END_STRUCT;
END_TYPE
```

Note

ARRAY OF BOOL are always sent to even-numbered addresses. For this reason, an array range of 0 to 15 must generally be selected in the UDT definition or all Boolean variables specified individually.

Although only a REAL value is used initially in the example, a field (with one element) has been created for the variable. This ensures that extensions can be made easily in the future without the symbolic address being modified.

Symbolic accesses

An entry is made in the symbol table to allow data access in symbolic form:

Symbol	Operand	Data type
UData	DB 20	UDT 20

Access operations in user program (list includes only symbolic read access):

```
...
          L       "UData".UDInt[0];
          L       "UData".UDInt[1];
          L       "UData".UDInt[2];
          L       "UData".UDInt[3];

          U       "UData".UDHex0[0];
          U       "UData".UDHex0[1];
          U       "UData".UDHex0[2];
          U       "UData".UDHex0[3];
          U       "UData".UDHex0[4];
          U       "UData".UDHex0[5];
          U       "UData".UDHex0[6];
          U       "UData".UDHex0[7];
          ...     ...
          U       "UData".UDHex0[15];

          L       "UData".UDReal[0];
...
```

12.10.11 Configuration machine control panel, handheld unit, direct keys

General

Up to two machine control panels and one handheld unit can be in operation at the same time. There are various connection options (Ethernet, PROFIBUS) for the machine control panel (MCP) and handheld unit (HHU). It is possible to connect two MCPs to different bus systems (mixed operation is only possible on Ethernet and PROFIBUS). This can be achieved using FB1 parameter "MCPBusType". In this parameter, the right-hand decade is responsible for the 1st MCP and the left-hand decade for the 2nd MCP.

The components are parameterized by calling basic-program block FB 1 in OB 100. FB 1 stores its parameters in the associated instance DB (DB 7, symbolic name "gp_par"). Separate parameter sets are provided for each machine control panel and the handheld unit. The input/output addresses of the user must be defined in these parameter sets. These input/output addresses are also used in FC 19, FC 24, FC 25, FC 26 and FC 13. Further, the addresses for status information, PROFIBUS or Ethernet are also to be defined. The default time settings for timeout and cyclic forced retriggering should not be changed. Please refer to the Operator Components manual for further information on MCPs and handheld unit components.

Activation

Each component is activated either via the number of machine control panels (MCPNum parameter) or, in the case of the handheld unit, via the HHU parameter. The MCP and HHU connection settings are entered in FB1 parameters "MCPMPI", "MCPBusType" or "BHG", "BHGMPI".

Handheld unit (HT2)

In the handheld unit the addressing is done via a parameter of the GD parameter set. This was necessary for reasons of compatibility of the parameter names.

Configuration

Essentially, there are various communication mechanisms for transferring data between the MCP/HHU and PLC. These mechanisms are characterized by the bus connection of the MCP and HHU. In one case (Ethernet), data is transported via the "CP 840D sl".

The mechanism is parameterized completely via the MCP/HHU parameters in FB1.

In the other case the transmission is via the PLC operating system through the PROFIBUS configuration

The parameter setting is done via STEP 7 in HW-Config. To enable the basic program to access the data and failure monitoring of MCP/HHU, the addresses set in FB 1-parameters must be made available to the basic program.

An overview of the various coupling mechanisms is shown below. Mixed operation can also be configured.

If an error is detected due to a timeout monitor, an entry is made in the alarm buffer of the PLC CPU (alarms 400260 to 400262). In this case, the input signals from the MCP or from the handheld unit (MCP1In/MCP2In or BHGIn) are reset to 0. If it is possible to resynchronize the PLC and MCP/HHU, communication is resumed automatically and the error message reset by the GP.

Note

The abbreviation "(n.r.)" in the tables below means "Not relevant".

Ethernet connection

Without further configuration settings being made, communication takes place directly from the PLC GP via the CP 840D sl. The FB 1 parameters listed below are used for parameterization.

The numeric part of the logical name of the component must be entered in "MCP1 BusAdr", "MCP2 BusAdr" or "BHGRecGDNo" (corresponds to the bus address of the node). The logical name is defined via switches on the MCP or terminal box.

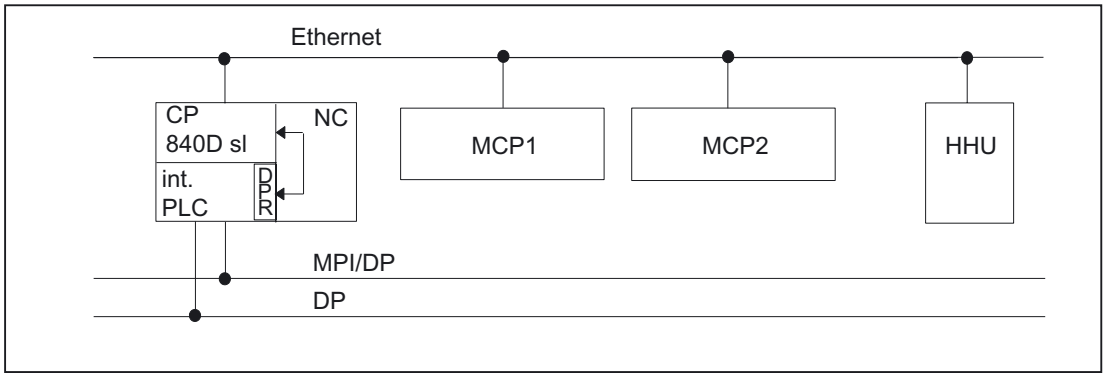


Figure 12-14 Ethernet connection

Relevant parameters (FB 1)		
MCP		**HHU**
MCPNum=1 or 2 (number of MCPs)		HHU = 5 (via CP 840D sl)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend	MCP2StatSend	BHGStatSend
MCP1StatRec (n.r.)	MCP2StatRec (n.r.)	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen (n.r.)
MCP1Timeout (n.r.)	MCP2Timeout (n.r.)	BHGOutLen (n.r.)
MCP1Cycl (n.r.)	MCP2Cycl (n.r.)	BHGTimeout (n.r.)
MCPMPI = FALSE		BHGCycl (n.r.)
MCP1Stop	MCP2Stop	BHGRecGDNo
MCP1 NotSend	MCP2 NotSend	BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPBusType = b#16#55 (via CP 840D sl)		BHGSendGDNo (n.r.)
		BHGSendGBZNo (n.r.)
MCPSDB210= FALSE		BHGSendObjNo (n.r.)
MCPCopyDB77 = FALSE		**BHGMPI = FALSE**
		BHGStop
		BHG NotSend

An error entry is also made in the PLC alarm buffer for timeouts. As a result, the following error messages are output at the HMI:

- 400260: MCP 1 failure

  or

- 400261: MCP 2 failure
- 400262: HHU failure

An MCP or HHU failure is detected immediately after a cold restart even if no data has yet been exchanged between the MCP/HHU and PLC.
The monitoring function is activated as soon as all components have signaled "Ready" after powerup.

Example: OP with direct keys

The direct keys of the OPs at the Ethernet Bus should be transferred to the PLC. Previously, the direct keys have been transferred to the PLC via the PROFIBUS or via a special cable connection between OP and MCP.

For connecting the direct keys via the Ethernet, this concerns e.g. the "OP08T", there is a parameterization in the basic program for activating the data transport. The related parameters lie in the instance DB of the FB 1 (OpKeyNum to OpKeyBusType, see data table). The parameters are provided by the user in the start OB 100 through the switching of the parameter at the FB 1 call. The bus address and Op1/2KeyStop can also be modified in the cyclic operation by writing the FB 1-Instance-DB DB 7.

The transport of the user data of the direct keys runs in the same way as in the case of Ethernet MCP. The data transport can also be stopped and restarted via writing the DB 7-parameter "Op1/2KeyStop". During the Stop phase the address of the direct key module (TCU-index or the MCP-address) can also be changed.

After resetting the Stop signal, a connection to the new address is established.

The status of the respective direct-key interfaces can be read in the interface signal:

DB10.DBX104.3 (OP1Key ready)

or

DB10.DBX104.4 (OP2Key ready)

Address direct keys

For the parameter Op1/2KeyBusAdr the TCU index is normally to be used. This affects the OPs, such as OP08T, OP12T, which for the direct keys **do not** have special cable connection to an Ethernet MCP.

If OPs with direct keys have a special cable connection and these are connected to an Ethernet-MCP, then for the parameter Op1/2KeyBusAdr the address of the MCP (DIP-switch setting of the MCP) is to be used. Only the data stream of the direct keys (2 bytes) is transferred via the direct key interface.

Alarm direct keys

An error entry is also made in the PLC alarm buffer for timeouts. As a result, the following error messages are output at the HMI:

- 400274: Direct key 1 failed

  or

- 400275: Direct key 2 failed

Control unit switching for direct keys

The user switches Op1/2KeyBusAdr with 0xFF and Stop = TRUE in the startup block OB 100. Via the M to N block FB 9 the direct key address of the M to N-interface is stored to the parameter "Op1KeyBusAdr".

Relevant parameters (FB 1)		
Direct keys		**e.g. direct keys OP08T**
OpKeyNum = 1 or 2 (number of OPs with direct keys)		
Op1KeyIn	Op2KeyIn	
Op1KeyOut	Op2KeyOut	
OpKey1BusAdr	Op2KeyBusAdr	Address: TCU index:
Op1KeyStop	Op2KeyStop	
Op1KeyNotSend	Op2KeyNotSend	
OpKeyBusType = b#16#55 (via CP 840D sl)		

MCP identification

Via the Identify interface in the DB 7 it is possible to query the type of the Ethernet component (MCP, HT2, HT8 or direct keys) with the relevant parameters at the input/output in cyclic operation:

- Relevant parameters at the input:

  "IdentMcpBusAdr", "IdentMcpProfilNo", "IdentMcpBusType", "IdentMcpStrobe"

- Relevant parameters at the output:

  "IdentMcpType", "IdentMcpLengthIn", "IdentMcpLengthOut"

Here the DIP device address or the TCU index at the parameter "IdentMcpBusAdr" is activated by the user program together with setting of the Strobe signal.

The input parameter "IdentMcpProfilNo" is normally to be set to the value 0. This parameter is to be set to the value 1 only in the identification of the direct keys. The parameter "IdentMcpBusType" currently has no significance for a user program and is to be left in its default value.

After resetting the Strobe signal by the basic program, valid output information becomes available to the user. The resetting of the Strobe signals by the basic program can last for several PLC cycles (up to two seconds).

The output parameters should show the user the size of the data areas for the addressed device. Furthermore, it can be defined here, whether an HT2 or an HT8 or no device is connected to the connection box. With this information, the MCP channel or the HHU-channel can be activated. In the cyclic operation the parameters can be written symbolically by the user program and read via the symbol names of the DB 7 (gp_par).

Relevant parameters (FB 1)		
MCP device identification		**Input parameters, e.g. OP08T**
Input	**Output**	**Values in direct keys**
IdentMcpBusAdr	IdentMcpType	IdentMcpBusAdr = TCU-Index
IdentMcpBusProfilNo	IdentMcpLengthIn	IdentMcpBusProfilNo = Value 1
IdentMcpBusType	IdentMcpLengthOut	IdentMcpBusType = Default value
IdentMcpStrobe		
IdentMcpBusProfilNo		**Value**
MCP, BHG, HT8, HT2		B#16#0
Direct keys, such as OP08T, OP12T		B#16#1
IdentMcpType (Mcp-Type)		
no device connected		0
MCP 483C IE (Compact)		B#16#80
MCP 483C IE		B#16#81
MCP 310		B#16#82
MCP OEM		B#16#83
MCP DMG		B#16#84
HT8		B#16#85
TCU_DT (direct keys)		B#16#86
MCP_MPP		B#16#87
HT2		B#16#88
OP08T (direct keys)		B#16#89

PROFIBUS connection on the DP port

In case of PROFIBUS connection of the MCP, this component must be considered in the hardware configuration setting of STEP 7. The MCP is connected to the standard DP bus of the PLC (**not to MPI/DP**). The addresses must be stored in the input and output log range. These start addresses must also be stored in the pointer parameters of the FB1. The FB1 parameters listed below are used for further parameterization.

There is no PROFIBUS variant of the HHU. For this reason, an Ethernet connection is shown for the HHU in this figure. The PROFIBUS slave address must be stored in the parameters "MCP1BusAdr" and "MCP2BusAdr". Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in "MCPxStatRec".

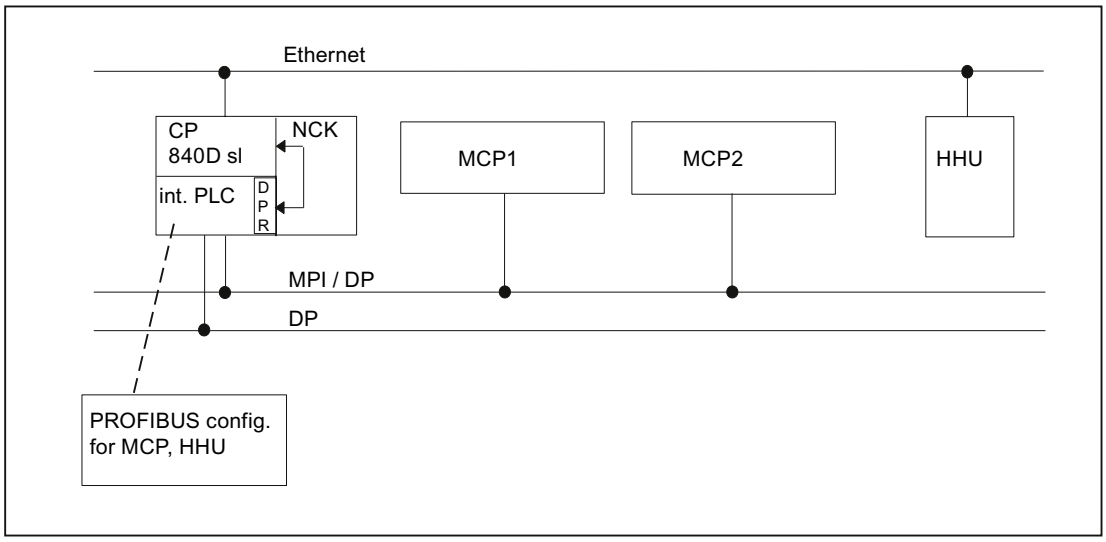


Figure 12-15 PROFIBUS connection

Relevant parameters (FB 1)		
MCP		**HHU**
MCPNum = 1 or 2 (number of MCPs)		HHU = 5 (via CP 840D sl)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend (n.r.)	MCP2StatSend (n.r.)	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl (n.r.)	MCP2Cycl	BHGTimeout (n.r.)
MCPMPI = FALSE		BHGCycl (n.r.)
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBusType = b#16#33		BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPSDB210= FALSE		BHGSendGDNo (n.r.)
MCPCopyDB77 = FALSE		BHGSendGBZNo (n.r.)
		BHGSendObjNo (n.r.)
		BHGMPI = FALSE
		BHGStop

MCP failure normally switches the PLC to the STOP state. If this is undesirable, OB 82, OB 86 can be used to avoid a stop. The basic program has, as standard, the OB 82 and OB 86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. Setting "MCPxStop" := TRUE causes the basic program to deactivate the MCP as a slave via SFC 12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The interrupt is deleted when the station recovers.

PROFIBUS connection on the MPI/DP port

With the PROFIBUS connection of the MCP, this component must be considered in the STEP 7 hardware configuration. The MCP is connected on the MPI/DP bus of the PLC.

The addresses must be stored in the input and output log range. These start addresses must also be stored in the pointer parameters of the FB1. The FB1 parameters listed below are used for further parameter assignment. There is no PROFIBUS variant of the HHU. For this reason, an Ethernet connection is shown for the HHU in this diagram. The PROFIBUS slave address must be stored in the parameters MCP1BusAdr and MCP2BusAdr. Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in MCPxStatRec.

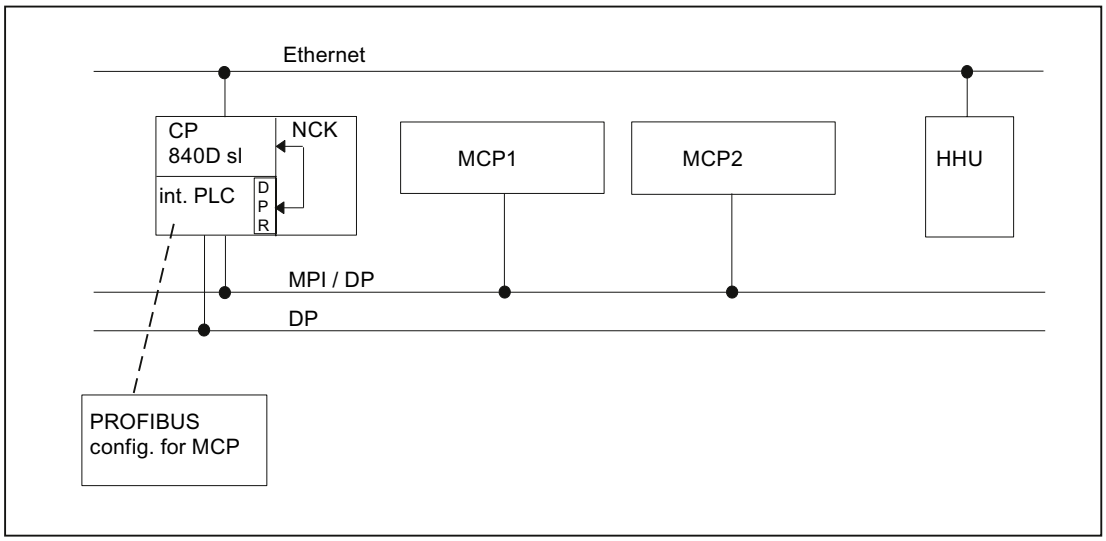


Figure 12-16 PROFIBUS connection on the MPI/DP port

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum = 1 or 2 (number of MCPs)		HHU = 5 (via CP 840D sl)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend (n.r.)	MCP2StatSend (n.r.)	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl (n.r.)	MCP2Cycl	BHGTimeout (n.r.)
MCPMPI = FALSE		BHGCycl (n.r.)
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBusType = b#16#44		BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPSDB210= FALSE		BHGSendGDNo (n.r.)
MCPCopyDB77 = FALSE		BHGSendGBZNo (n.r.)
		BHGSendObjNo (n.r.)
		BHGMPI = FALSE
		BHGStop

MCP failure normally switches the PLC to the STOP state. If this is not wanted, OB 82, OB 86 can be used to avoid a PLC stop. The basic program has, as standard, the OB 82 and OB 86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. Setting MCPxStop := True causes the basic program to deactivate the MCP as a slave via SFC 12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The alarm is deleted when the station returns.

PROFINET connection

In case of a PROFINET connection of the MCP, this component must be parameterized in the hardware configuration of STEP 7. The MCP is coupled with the PROFINET module of the CPU.

When parameterizing the MCP in HW Config, the addresses should be placed in the input and output mapping area. These start addresses must also be stored in the pointer parameters (MCPxIn and MCPxOut) of FB1. This is because signals are transferred between the MCP and basic program via these parameters. The MCP is also monitored using parameter MCPxIn. This is the reason why parameter MCPxBusAdr is not relevant for this MCP version.

Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in MCPxStatRec.

The PROFINET-MCP has its own type which should be applied for parameter MCPBusType.

The FB1 parameters listed below are used for further parameter assignment. There is no PROFIBUS variant of the HHU. An Ethernet port for the HHU is shown in the diagram.

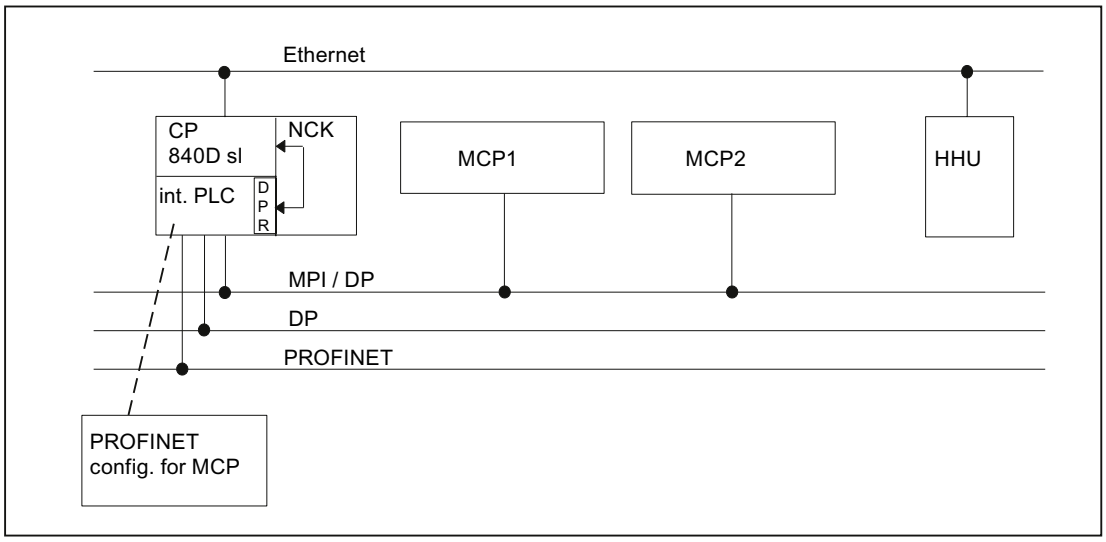


Figure 12-17 PROFINET connection

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum = 1 or 2 (number of MCPs)		HHU = 5 (via CP 840D sl)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend (n.r.)	MCP2StatSend (n.r.)	BHGStatSend
MCP1StatRec (n.r.)	MCP2StatRec (n.r.)	BHGStatRec
MCP1BusAdr (n.r.)	MCP2BusAdr (n.r.)	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl	MCP2Cycl	BHGTimeout (n.r.)
MCPMPI = FALSE		BHGCycl (n.r.)
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBusType = b#16#36		BHGRecGBZNo (n.r.)
		BHGRecObjNo (n.r.)
MCPSDB210= FALSE		BHGSendGDNo (n.r.)
MCPCopyDB77 = FALSE		BHGSendGBZNo (n.r.)
		BHGSendObjNo (n.r.)
		BHGMPI = FALSE
		BHGStop

MCP failure normally switches the PLC to the STOP state. If this is not wanted, then OB 82, OB 86 can be used to avoid a PLC stop. The basic program has, as standard, the OB 82 and OB 86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. The input address at parameter MCPxIn is of significance when monitoring for MCPxIn failure.

Setting MCPxStop := True causes the basic program to deactivate the MCP as a slave via SFC 12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The alarm is deleted when the station returns.

12.10.12 Switchover of machine control panel, handheld unit

Only Ethernet variants support switchover/deactivation of the machine control panel (MCP) or handheld unit (HHU) as standard. On MPI and PROFIBUS variants, this function is either not supported at all or can only be implemented in restricted format requiring additional effort on the part of the user. For example, with the PROFIBUS variant of the MCP, the DB77 data area specified for MCP1, MCP2 or HHU can be used for the MCP pointer on FB1. The MCP slave bus address must be set correctly under MCPxBusAdr as this is used as the basis for monitoring. A user program copy routine to copy the signals of the active MCP from the I/O area configured in HW Config to DB77. This enables a number of MCPs on the PROFIBUS to be switched via signals. Set the MCPxStop parameter to True for the switchover phase from one MCP to another.

One method will be outlined now with the Ethernet variants of MCP and HHU.

Control signals

Parameters MCP1Stop, MCP2Stop and BHGStop can be used to stop communication with individual components (parameter setting = 1). This function is available only on Ethernet variants. This stop or activation of communication can be applied in the current cycle. However, the change in value must be implemented through the symbolic notation of the parameters and not by means of another FB 1 call.

Example of stopping transfer from the 1st machine control panel:

```
SET;
S gp_par.MCP1Stop;
```

Setting parameters MCP1Stop, MCP2Stop, BHGStop also results in a suppression or deletion of interrupts 400260 to 400262.

Switchover of Bus address

An existing connection with a machine control panel (MCPl) or handheld unit (HHU) can be aborted. Another MCP or HHU component already connected to the bus (different address) can then be activated. Proceed as follows to switch addresses:

1. Stop communication with component to be decoupled via parameter MCP1Stop or MCP2Stop or BHGStop = 1.
2. Following checkback in DB10 byte 104 (relevant bits 0, 1 and 2 are set to 0), the bus address (with MCP, this is the FB1 parameter "MCP1BusAdr" or "MCP2BusAdr") is changed; With HHU Ethernet variant, the bus address is set at FB1 parameter "BHGRecGDNo") of this unit is changed to the new component.
3. In this PLC cycle, communication with the new component can now be activated again by means of parameter MCP1Stop or MCP2Stop or BHGStop = 0.
4. Communication with the new component is taking place when the checkback is in DB 10 byte 104 (relevant bits 0, 1, 2 are set to 1).

Switching off the MCP IE flashing

General behavior and general conditions

After POWER ON the MCPs always flash and indicate the completion of the power-up sequence and that the system is waiting for a connection to be established (default setting of the MCP).

When establishing a connection to the MCP (e.g. the PLC connects itself to the appropriate MCP) the behavior of the LEDs can be set for the subsequent state where there is no communication.

Presently, this behavior cannot be retentively stored on the MCP!

Setting via the PLC:

With MCP firmware V02.02.04 and higher, flashing can be suppressed in the offline mode.

No communication takes place in the offline mode (e.g. even if the MCP connection fails).

In order to deactivate the flashing, the bits in FB1 parameter MCPxStatSend (status double word for sending) must be set as follows before communication to the MCP starts:

Bit 30 = FALSE and bit 31 = TRUE

Before communication starts to the MCP means:

- In OB100 or
- In OB1 before MCP stop/start

  (DB7 parameter change MCPxStop = TRUE → MCPxStop = FALSE)

Feedback of the status is presently not available.

Example:

```
Extract from OB100: (based on the example for MCP1)

CALL "RUN_UP" , "gp_par"
...
MCP1StatSend            := P#A 8.0
...

//deactivate MCP flashing
SET
R      A 11.6
S      A 11.7
...
```

12.11 SPL for Safety Integrated

Rather than being a function of the basic program, SPL is a user function. The basic program makes a data block (DB 18) available for Safety SPL signals and runs a data comparison to ensure the consistency of SPL program data in the NCK.

References:
/FBSI/ Description of Functions Safety Integrated

12.12 Assignment overview

12.12.1 Assignment: NCK/PLC interface

The values of the NC/PLC interface for SINUMERK 840D sl are described in detail in:
References:
Lists sl (Buch2)

12.12.2 Assignment: FB/FC

Number	Significance
FB 15	Basic program
FB 1, FC 2, FC 3, FC 5	Basic program
FC 0 ... 29	Reserved for Siemens
FB 0 ... 29	Reserved for Siemens
FC 30 ... 999 [1)]	Free for user assignment
FB 30 ... 999 [1)]	Free for user assignment
FC 1000 ... 1023	Reserved for Siemens
FB 1000 ... 1023	Reserved for Siemens
FC 1024 ... upper limit	Free for user assignment
FB 1024 ... upper limit	Free for user assignment

[1)] The actual upper limit of the block number (FB/FC) depends on the PLC CPU on which the selected NCU is located.

Note

Values of FC, FB see " Memory requirements of the basic PLC program (Page 979)".

12.12.3 Assignment: DB

Note

Only as many data blocks as are required according to the NC machine data configuration are set up.

Overview of data blocks			
DB no.	**Name**	**Name**	**Package**
1		Reserved for Siemens	BP
2 ... 5	PLC-MELD	PLC messages	BP
6 ... 8		Basic program	
9	NC-COMPILE	Interface for NC compile cycles	BP
10	NC INTERFACE	Central NCK interface	BP
11	Mode group 1	Interface mode group	BP
12		Computer link and transport system interface	
13 ... 14		Reserved for basic program	
15		Basic program	
16		PI Service definition	
17		Version identifier	
18		Reserved for basic program	
19		HMI interface	
20		PLC machine data	
21 ... 30	CHANNEL 1 ... n	Interface NC channels	BP
31 ... 61	AXIS 1 ... m	Interfaces for axes/spindles or free for user assignment	BP
62 ... 70		Free for user assignment	
71 ... 74		Tool management	BP
75 ... 76		M group decoding	
77		Data block for MCP signals	
78 ... 80		Reserved for Siemens	
81 ... 999 [1)]		see below: ShopMill, ManualTurn	
1000 ... 1099		Reserved for Siemens	
1100 ... High limit		Free for user assignment	

[1)] The actual upper limit of the block number (DB) depends on the PLC CPU on which the selected NCU is located. The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as desired by the user.

Note

The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as required by the user.

12.12.4 Assignment: Timers

Timer No.	Significance
T 0 ... T 512 [1)]	User area

[1)] The actual upper limit of the timer number (DB) depends on the PLC CPU on which the selected NCU is located.

12.13 Memory requirements of the basic PLC program

The basic program consists of basic and optional functions. The **basic functions** include cyclic signal exchange between the NC and PLC. The **Options** include e.g. the FCs, which can be used, if needed.

The table below lists the memory requirements for the basic functions and the options. The data quoted represent guide values, the actual values depend on the current software version.

Memory requirements of blocks for SINUMERIK 840D sl

Block type no.	Function	Remark	Block size (bytes) Working memory
Basic functions in basic program			
FB 1, FB 15		must be loaded / on CF card	52182
FC 2, 3, 5, 12		Must be loaded	470
DB 4, 5, 7, 8		Must be loaded	1006
DB 2, 3, 17		Are generated by the BP	632
OB 1, 40, 100, 82, 86		Must be loaded	398
		Total	55698
PLC/NCK, PLC/HMI interface			
DB 10	PLC/NCK signals	Must be loaded	262
DB 11	Signals PLC/Mode group	Is generated by BP	56
DB 19	PLC/HMI signals	Is generated by BP	434
DB 21 to DB 30	PLC/channel signals	Are generated by BP as a function of NCMD: for each DB	416
DB 31 to DB 61	PLC / axis or spindle signals	Are generated by BP as a function of NCMD: for each DB	148

Basic program options			
	Machine control panel		
FC 19	Transfer of MCP signals, M variant	Must be loaded when M variant of MCP is installed	92
FC 25	Transfer of MCP signals, T variant	Must be loaded when T variant of MCP is installed	92
FC 24	Transfer of MCP signals, slim variant	Must be loaded when slim variant of MCP is installed	100
FC 26	Transfer of MCP signals, HT8 variant	Must be loaded for HT8	68
	Handheld unit		
FC 13	Display control HHU	Can be loaded for handheld units	144
	Error/operating messages		
FC 10	Acquisition FM/BM	Load when FM / BM is used	66
	ASUB		
FC 9	ASUB start	Load when PLC ASUBs are used	128

Basic program options			
	Star/delta changeover		
FC 17	Star/delta switchover of MSD	Load for star/delta switchover	114
	Spindle control		
FC 18	Spindle control	Load for spindle control from PLC	132
	PLC/NC communication		
FB 2	Read NC variable	Load for Read NC variable	76
DB n	Read NC variable	One instance DB per FB 2 call	270
FB 3	Write NC variable	Load for Write NC variable	76
DB m	Write NC variable	One instance DB per FB3 call	270
FB 4	PI services	Load for PI services	76
DB o	PI services	One instance DB per FB 4 call	130
DB 16	PI services description	Load for PI services	618
FB 5	Read GUD variables	Load for PI services	76
DB p	Read GUD variables	One instance DB per FB 5 call	166
DB 15	General communication	Global data block for communication	146
FB 7	PI services 2	Load for PI services	76
DB o	PI services 2	One instance DB per FB4 call: every	144
FC 21	Transfer	Load with dual-port RAM, ...	164
	m:n		
FB 9	Switchover M to N	Load with M to N	58
	Safety Integrated		

Basic program options			
FB 10	Safety relay	Load with Safety option	74
FB 11	Brake test	Load with Safety option	76
DB 18	Safety data	DB for Safety	308
	Tool management		
FC 7	Transfer function turret	Load for tool management option	84
FC 8	Transfer function	Load for tool management option	132
FC 22	Direction selection	Load, when direction selection is needed	138
DB 71	Loading locations	Generated by BP as a function of NC MD	40+30*B
DB 72	Spindles	Generated by BP as a function of NC MD	40+48*Sp
DB 73	Revolver	Generated by BP as a function of NC MD	40+44*R
DB 74	Basic function	Generated by BP as a function of NC MD	100+(B+ Sp+R)*22
	Compile cycles		
DB 9	Interface PLC compile cycles	Is generated by BP as a function of NC option	436

Example:

Based on the memory requirements in the table above, the memory requirements have been determined for two sample configurations (see table below).

Block type no.	Function	Remark	Block size (bytes) Working memory
Minimum configuration (1 spindle, 2 axes and T MCP)			
see above	Basic program, base		54688
	Interface DBs		1612
	MCP		92
		Total	56392

Block type no.	Function	Remark	Block size (bytes) Working memory
Maximum configuration (2 channels, 4 spindles, 4 axes, T MCP)			
see above	Basic program, base		54688
see above	Interface DBs		2768
see above	MCP		92
see above	Error/operating messages		66
see above	ASUBs	1 ASUB initiation	128
see above	Concurrent axis	For 2 turrets	132
see above	PLC/NC communication	1 x read variable and 1 x write variable	838
see above	Tool management	2 turrets with one loading point each	674
see above	Compile cycles		436
		Total	59822

12.14 Basic conditions and NC VAR selector

12.14.1 Supplementary conditions

12.14.1.1 Programming and parameterizing tools

Hardware

For the PLCs used in SINUMERIK 840D sl, the following equipment is required for the programming devices or PCs:

	Minimum	Recommendation
Processor	Pentium	Pentium
RAM (MB)	256	512 or more
Hard disk, free capacity (MB)	500	> 500
Interfaces	MPI, Ethernet incl. cable Memory card	
Graphic	SVGA (1024*768)	
Mouse	Yes	
Operating system	Windows 2000 /XP Professional, STEP 7 version 5.3 SP2 or higher	

The required version of **STEP 7** can be installed on equipment meeting the above requirements in cases where the package has not already been supplied with the programming device.

The following functions are possible with this package:

- Programming
  - Editors and compilers for STL (complete scope of the language incl. SFB/SFC calls), LAD, FBD
  - Creation and editing of assignment lists (symbol editor)
  - Data block editor
  - Input and output of blocks ON/OFF line
  - Insertion of modifications and additions ON and OFF line
  - Transfer of blocks from programming device to the PLC and vice versa
- Parameterizing
  - Parameterizing tool **HW Config** for CPU and I/O device parameterization
  - **NetPro** parameterizing tool for setting the CPU communication parameters
  - Output of system data such as hardware and software version, memory capacity, I/O expansion/assignment

- Testing and diagnostics (ONLINE)
  - Variable status/forcing (I/Os, flags, data block contents, etc.)
  - Status of individual blocks
  - Display of system states (ISTACK, BSTACK, system status list)
  - Display of system messages
  - Trigger PLC stop / restart / general reset from the PG
  - Compress PLC
- Documentation
  - Printout of individual or all blocks
  - Allocation of symbolic names (also for variables in data blocks)
  - Input and output of comments within each block
  - Printout of test and diagnostics displays
  - Hardcopy function
  - Cross-reference list
  - Program overview
  - Assignment plan I/O/M/T/Z/D
- Archiving of utility routines
  - Allocation of the output states of individual blocks
  - Comparison of blocks
  - Rewiring
  - STEP 5 → STEP 7 converter
- Option packages
  - Programming in S7-HIGRAPH, S7-GRAPH, SCL.

    These packages can be ordered from the SIMATIC sales department.
  - Additional packages for configuration modules (e.g. CP3425 → NCM package)

Note

More information about possible functions can be found in SIMATIC catalogs and STEP 7 documentation.

12.14.1.2 SIMATIC documentation required

ReferenceS:

- System description SIMATIC S7
- S7-300 instruction list
- Programming with STEP 7
- User Manual STEP 7
- Programming manual STEP 7; designing of user programs
- Reference manual STEP 7; Instructions list AWL
- Reference manual STEP 7; Ladder Diagram KOP
- Reference manual STEP 7; Default and system functions
- Manual STEP 7; Conversion of STEP 5 programs
- STEP 7 overall index
- Manual CPU 317-2DP

12.14.1.3 Relevant SINUMERIK documents

References:

- Commissioning Manual IBN CNC: NCK, PLC, Drive
- Operator Components and Networking Manual
- Function Manual Basic Functions
- Function Manual, Extended Functions:
- Function Manual, Special Functions
- Lists sl (Book1)
- Lists sl (Book2)

12.14.2 NC VAR selector

12.14.2.1 Overview

General

The PC application "NC VAR selector" fetches the addresses of required NC variables and processes them for access in the PLC program (FB 2/FB 3). This enables the programmer to select NC variables from the entire range of NC variables, to store this selection of variables, to edit them by means of a code generator for the STEP 7 compiler and finally to store them as an ASCII file (*.AWL) in the machine CPU program. This process is shown in the figure "NC VAR selector".

For storing the files created by NC-VAR-selector a catalog is to be implemented via the Windows Explorer with any catalog name. The selected data of the NC-VAR selector (data.VAR and data.AWL files) must be stored in this catalog. Thereafter, the STL file is to be transferred and compiled via the menu option "Code" → "in STEP 7 Project". The "data.AWL" (STL data) file must then be inserted into the STEP 7 machine project via "Insert", "External Source" in the STEP 7 Manager. The source container must be selected in the manager for this purpose. This action stores this file in the project structure. Once the file has been transferred, these AWL (STL) files must be compiled with STEP 7.

Note

The latest NC VAR selector can be used for each NC software version (even earlier versions). For older NC software versions the variables can also be selected from the latest complete list. The data content in DB 120 (default DB for variables) does not depend on the software status. That is, variables selected in an older software version need not be reselected when the software is upgraded.

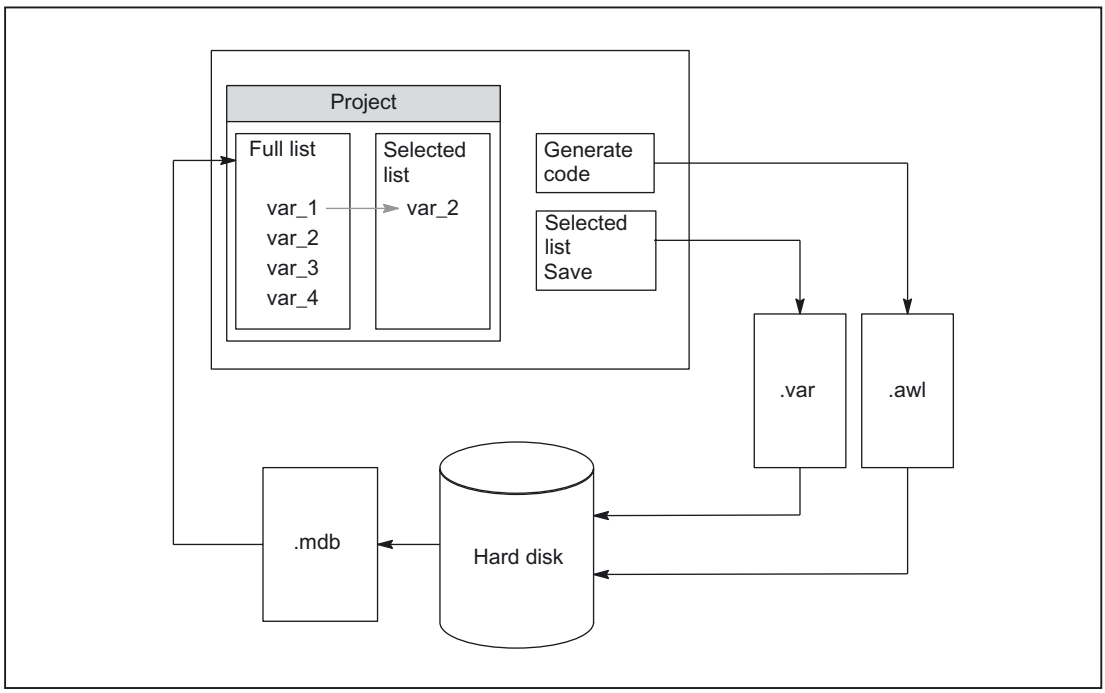


Figure 12-18 NC VAR selector

After the "NC VAR selector" application has been started, select a list of variables of an NC variant (hard disk → file Ncv.mdb) to display all the variables contained in this list in a window.

The following ncv*.mdb variable list is available:

Variables	List
NC variables including machine and setting data:	ncv_NcData.mdb
Parameters of the drive:	ncv_SinamicsServo.mdb

The user can also transfer variables to a second list (separate window). This latter selection of variables can then be stored in an ASCII file or edited as a STEP 7 source file (.awl) and stored.

After generating a PLC data block by means of the STEP 7 compiler, the programmer is able to read or write NCK variables via the basic program function blocks "PUT" and "GET" using the STEP 7 file.

The list of selected variables is also stored as an ASCII file (file extension .var).

The variable list supplied with the "NC VAR selector" tool is adapted to the current NC software version. This list does not contain any variables (GUD variables) defined by the user. These variables are processed by the function block FB 5 in the basic program.

Note

The latest version of the "NC VAR selector" is capable of processing all previous NC software versions. It is, therefore, not necessary to install different versions of the "NC VAR selector" in parallel.

System features, supplementary conditions

The PC application "NC VAR selector" requires Windows 2000 or a higher operating system.

The assignment of names to variables is described in:
References:
/LIS1/ Lists (Book1); Section: Variables,
or in the Variables Help file (integrated in NC VAR selector).

12.14.2.2 Description of functions

Overview

The figure below illustrates how the NC VAR selector is used within the STEP 7 environment.

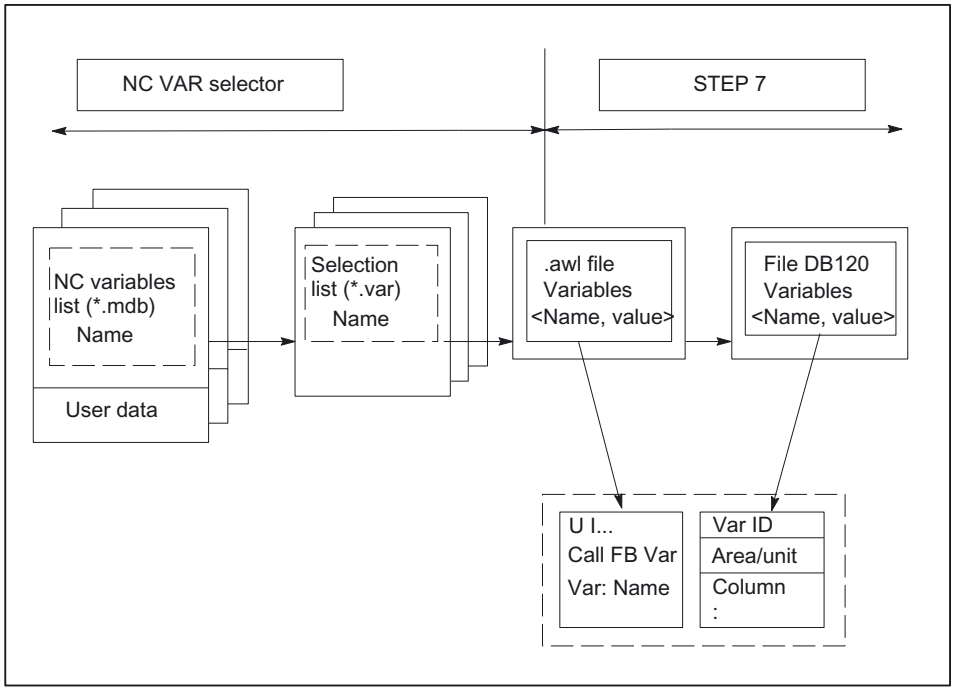


Figure 12-19 Application of NC VAR selector in the STEP 7 environment

The NC VAR selector is used to generate a list of selected variables from a list of variables and then to generate an **.awl** file that can be compiled by the STEP 7 compiler.

- A *.awl file contains the names and alias names of the NC variables, as well as information about their address parameters. Any data block generated from this file will only contain the address parameters (10 bytes per parameter).
- The generated data blocks must always be stored in the machine-specific file storage according to STEP 7 specifications.
- To ensure that the parameterization of the GET/PUT (FB 2/3) blocks with respect to NC addresses can be implemented with symbols, the freely assignable, symbolic name of the generated data block must be included in the STEP 7 symbol table.

Basic display / basic menu

After the NC VAR selector has been selected (started), the basic display with all input options (upper menu bar) appears on the screen. All other displayed windows are placed within the general window.

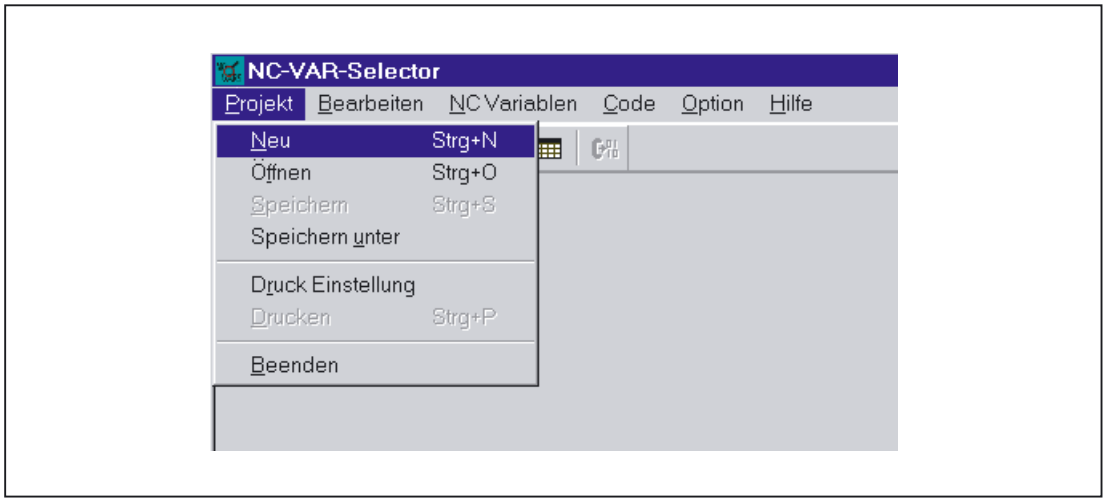


Figure 12-20 Basic display with basic menu

Project menu item

All operator actions associated with the project file (file of selected variables) are performed under this menu item.

Terminating the application

The application can be terminated by selecting the "End" option under the "Project" menu item.

Creating a new project

A new project (new file for selected variables) can be set up under the "Project" menu item.

A window is displayed for the selected variables when "NEW" is selected. The file selection for the NC variable list is then displayed after a prompt (applies only if the NC variable list is not already open).

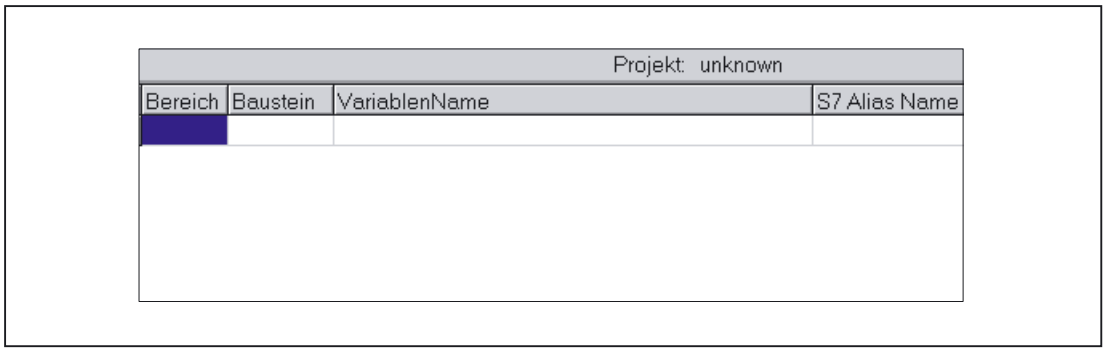


Figure 12-21 Window with selected variables for new project

The selected variables are displayed in a window.

Open an already existing project

Select "Open" under the "Project" menu item to open an existing project (variables already selected). A file selection window is displayed allowing the appropriate project with extension ".var" to be selected.

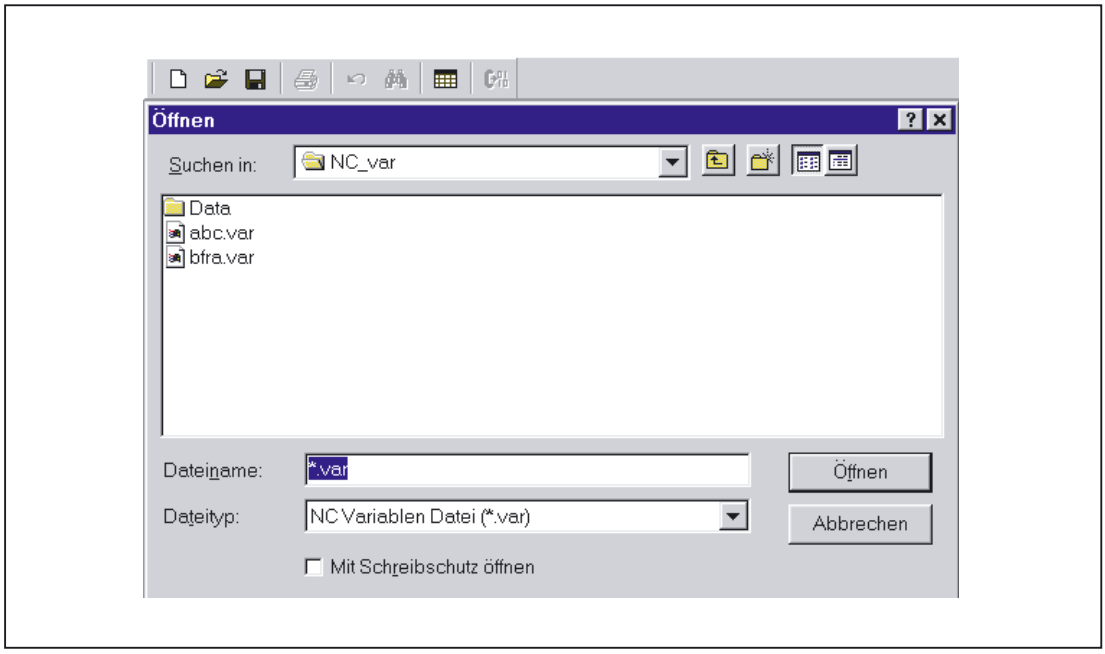


Figure 12-22 Selection window for existing projects

If, after selection of the project, new variables are to be added, a complete list of NC variables must be selected. No complete list need be called if the user only wishes to delete variables from the project.

Storing a project

The variable list is stored using the "Project" > "Save" or "Save As...." menu items.

"Save" stores the variable list under a path, which is already specified. If the project path is not known, then the procedure is as for "Save As....".

"Save As..." displays a window in which the path for the project to be stored can be specified.

Printing a project

The "Print" command under the "Project" menu item can be selected to print a project file. The number of lines per page is selected under the "Print Setting" menu item. The default setting is 77 lines.

Edit menu item

The following operator actions are examples of those, which can be carried out directly with this menu item:

- Transfer variables
- Delete variables
- change alias names
- Find variables

These actions can also be canceled again under Edit.

Undoing actions

Operator actions relating to the creation of the project file (transfer variables, delete variables, change alias names) can be undone in this menu.

NC variables menu item

The basic list of all variables is saved in NC Var Selector path Data\Swxy (xy stands for software version no., e.g. SW 5.3:=xy=53). This list can be selected as an NC variables list. In case of SINUMERIK 840D sl the basic lists are present in the path Data\Swxy_sl.

Selecting an NC variable list

A list of all the NC variables for an NC version can now be selected and displayed via the "NC Variable List", "Select" menu item.

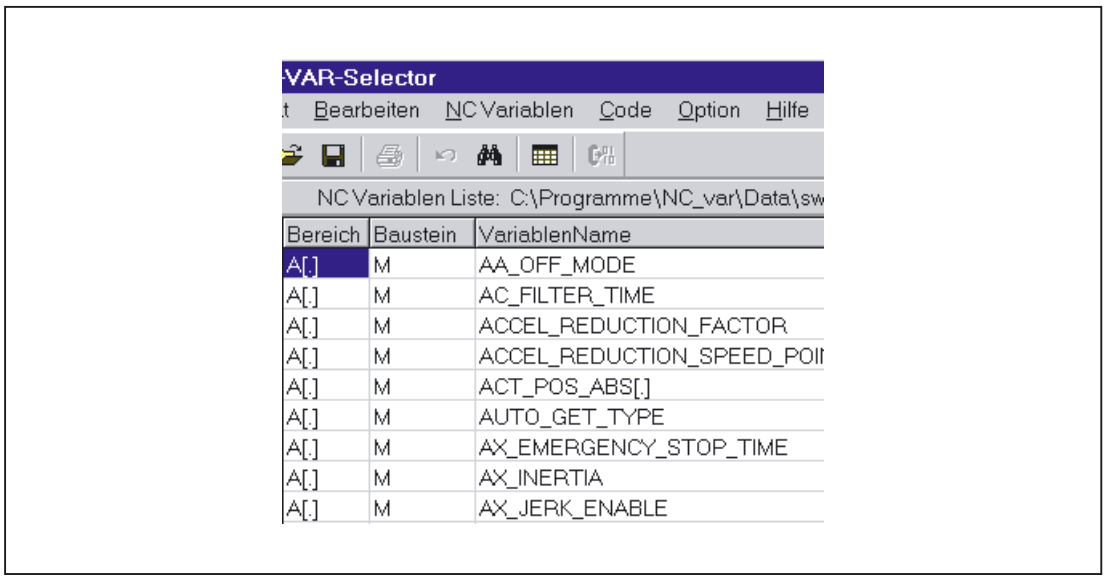


Figure 12-23 Window with selected Complete List

The field variables (e.g. axis area, T area data, etc.) are indicated by means of brackets ([.]). Additional information must be specified here. When the variables are transferred to the project list, the additional information required is requested.

Displaying subsets

Double-click on any table field (with the exception of variable fields) to display a window in which filter criteria can be preset.

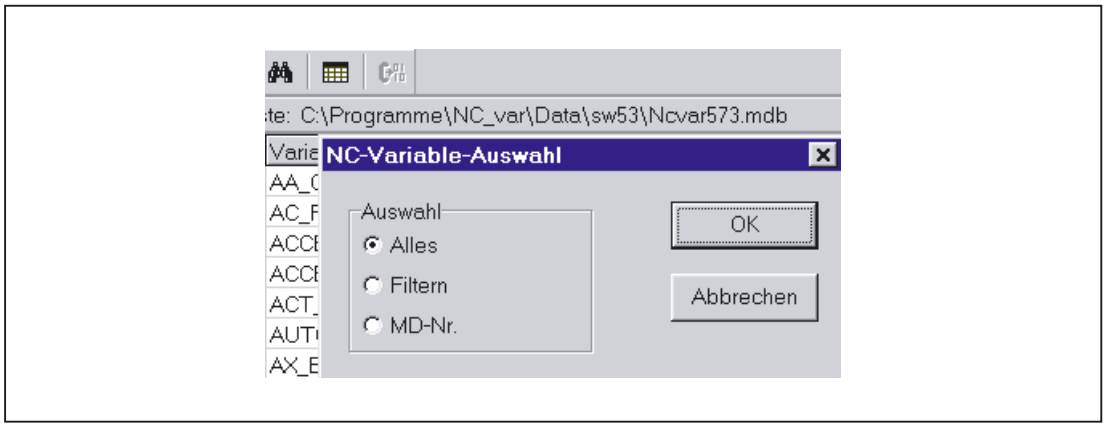


Figure 12-24 Window with filter criteria for displaying list of variables

There are three options:

- Display all data
- Input area, block and name (incl. combinations)
- Display MD/SE data number

The following wildcards can also be used:

*	To extend the search criterion as required

Example search criteria

Name search criterion: CHAN* Found:

CHAN_NAME
chanAlarm
chanStatus
channelName
chanAssignment

- Select variable

  A variable is selected by means of a simple mouse click and transferred to the window of selected variables by double-clicking. This action can also be undone under the "Edit" menu item.

Alias name

The variable names provided can be up to 32 characters in length. To make variables clearly identifiable in the data block to be generated, several ASCII characters are added to the selected name. However, the STEP 7 compiler recognizes only 24 ASCII characters as an unambiguous STEP 7 variable. Since it cannot be precluded that variable names can only be differentiated by the last 8 character positions, **ALIAS** names are used for names, which are too long. When a variable is selected, the length of the STEP 7 name to be used is, therefore, checked. If the name is longer than 24 characters, the user must enter an additional name, which is then used as the alias.

In this case, the user must ensure that the alias name is unambiguous.

Alias input can always be activated by the user in the "Options" menu.
An alias name can then be entered every time a variable is transferred.

It is also possible to edit alias names at a later point in time by double-clicking on the S7 variable name field. This action can also be undone under the "Edit" menu item.

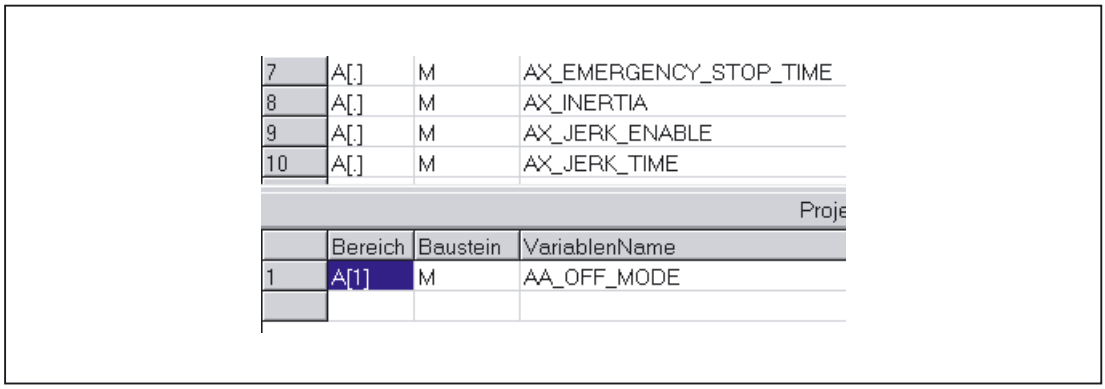


Figure 12-25 Screen with complete list and selected variables

Scrolling

A scroll bar is displayed if it is not possible to display all variables in the window. The remaining variables can be reached by scrolling (page up/down).

Variables in multi-dimensional structures

If variables are selected from multi-dimensional structures, then the column and/or line number as well as the area number must be entered so that the variables can be addressed. The required numbers can be found in the NC variables documentation.

References:
Lists sl (Book1); Variables

By entering a zero (0) as the block number or the line or column index, it is possible to use the variable in the S7 PLC as a pointer to these data. When reading or writing these data via the functions "PUT" and "GET", the optional parameters "UnitX", "ColumnX" and "LineX" must be filled with the necessary information.

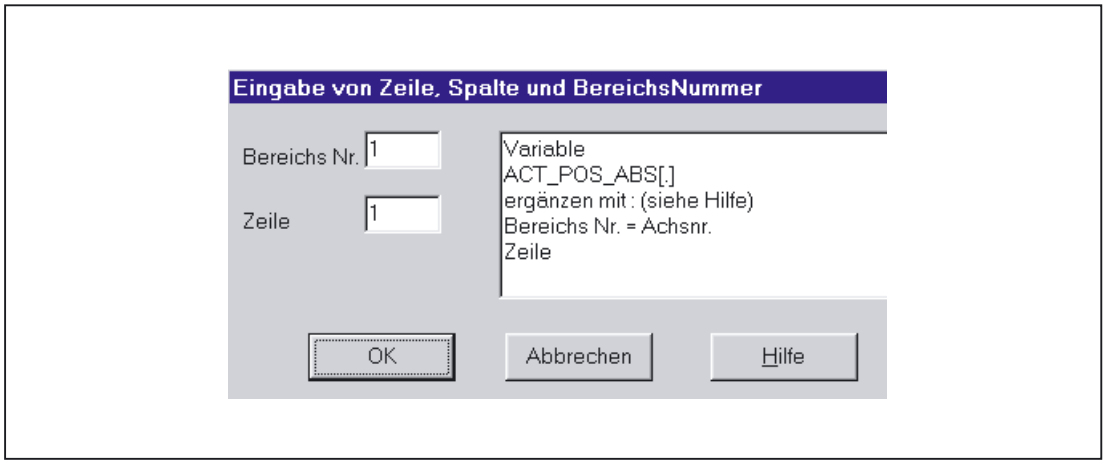


Figure 12-26 Entry field for line, column and block no.

Delete variables

Variables are deleted in the window of selected variables by selecting the appropriate variables (single mouse click) and pressing the "Delete" key. No deletion action is taken with the double-click function. It is possible to select several variables for deletion (see Section "Example of search criteria > Selecting variables").

This action can also be undone under the "Edit" menu item.

Note

Deleting of variables results in a change of the absolute addresses of the pointer structures to the variables. When changing the variable selection, it is, therefore, absolutely necessary to **generate** one or several **text files of all user blocksprior to the change**. This is the only way to ensure that the assignment of the variables in FB "GET" or FB "PUT" remains correct, even after recompilation.

Storing a selected list

Once variables have been selected, they can be stored under a project name. The files are stored on a project-specific basis.

A window is displayed for the file to be stored. The project path and name for the file must be selected in the window.

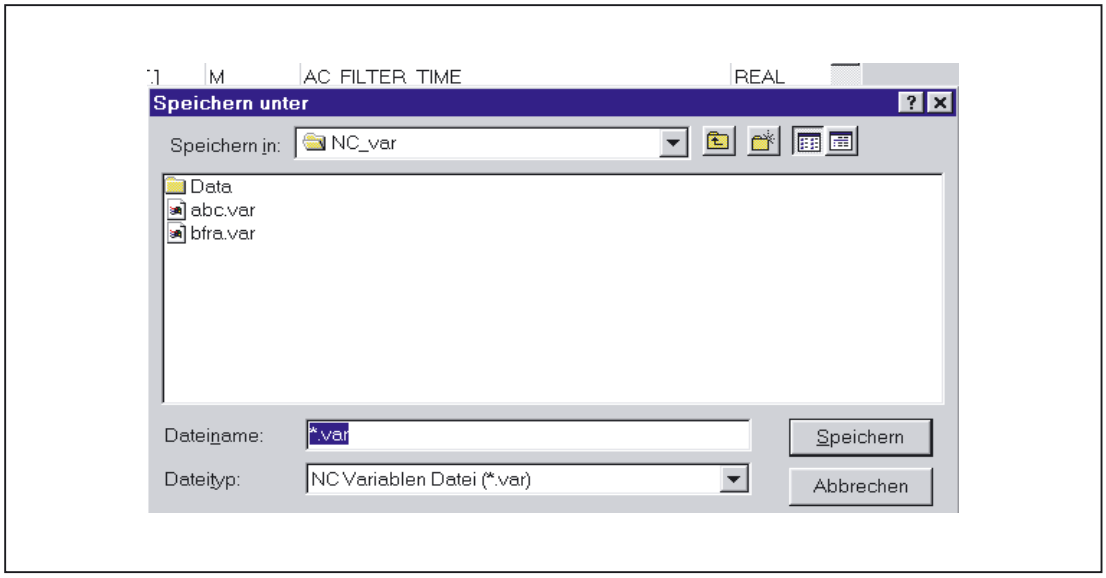


Figure 12-27 Window for project path and name of file to be stored

Code generation

This menu item contains three selection options:

1. Settings (input of data block number to be generated) and other settings
2. Generate (create data block)
3. In the STEP 7 project (transferring the data block to a STEP 7 project)

Settings

Under this menu item, the DB number and the symbol for this DB number for which the code is created is entered.

Under the "Mass System" tab, a selection is made to determine how the unit system variables are calculated in the PLC.

Under the "Generate" tab, the project creation is defined for the relevant target system.

Generate

Under this menu item, the STEP 7 file from the selected variable list with extension ".awl" is set.

A file is generated when "Select" is clicked:

An **.awl** file that can be used as an input for the STEP 7 compiler.

A window opens, in which path and name for the .awl file to be generated must be specified for the file to be saved.

In STEP 7 project

The generated STL file is transferred to a selectable SIMATIC project (program path) and compiled. Furthermore, the symbol can also be transferred. This function is available as of STEP 7 Version 5.1. This process takes a longer time owing to the call of STEP 7. Before transferring a new STL file the file window of the STL file is to be closed in the LAD/FBD/STL editor.

Option menu item

The following can be selected under the "Option" menu item:

- The current language
- The mode for alias input (always / > 24 characters)

Help menu item

The information below can be viewed by selecting the corresponding submenu item:

- The Operating Manual
- The Description of Variables

The copyright and the version number can also be displayed.

12.14.2.3 Startup, installation

The Windows application "NC Var selector" is installed using the SETUP program supplied with the package.

12.15 Block descriptions

12.15.1 FB 1: RUN_UP Basic program, startup section

Function

The synchronization of NCK and PLC is performed during startup. The data blocks for the NC/PLC user interface are created with reference to the NC configuration defined in the machine data and the most important parameters verified for plausibility. In the event of an error, FB 1 passes an error identifier to the diagnostics buffer and switches the PLC to the STOP state.

To enable an orderly start-up of the control, it is vital to synchronize the NCK and PLC, as these systems have their own types of power-up procedure. During startup routine, therefore, the CPUs perform "subsidiary startup functions" and exchange ID information to ensure that the procedure is functioning correctly.

Since the startup procedure is asynchronous, it is unavoidable that one CPU may have to "wait" until the other has "caught up". This is automatically managed by the basic program.

The integrated PLC only supports cold starts. A warm restart is not provided, i.e. following system initialization, the operating system runs organization block OB 100 and always commences cyclic execution at the start of OB 1.

Users need only supply the FB 1 parameters that are relevant to their applications. The preset values in the associated instance DB 7 do not need to be assigned. The block can only be called in OB 100.

Output parameters

The output parameters in FB 1 provide the PLC user with information about the control system configuration. These data can also be accessed in the cyclic program section.

There are two access options:

1. Direct access to the DB 7 data block (instance of the FB 1) in symbolic format (e.g. L gp_par.MaxChan; in this case, gp_par is the symbolic name of the DB 7)
2. Assignment of a flag; during parameterization of the FB 1, the data element is assigned to the relevant parameter (e.g. MaxChan:=MW 20) Information about the maximum number of channels can then be polled in memory word 20 in the rest of the user program.

Note

For the values of the parameters of MCP and HHU see "Configuration machine control panel, handheld unit, direct keys (Page 962) ".

Declaration SINUMERIK 840D sl

```
FUNCTION_BLOCK FB 1
VAR_INPUT
    MCPNum:                 INT:=1;                     //0: No MCP
                                                        //1: 1 MCP (default)
                                                        //2: 2 MCPs
    MCP1In:                 POINTER;                    //Start addr. input signals MCP 1
    MCP1Out:                POINTER;                    //Start addr. output signals MCP 1
    MCP1StatSend:           POINTER;                    //Status DW for sending MCP 1
    MCP1StatRec:            POINTER;                    //Status DW for receiving MCP 1
    MCP1BusAdr:             INT:=6;                     //Default
    MCP1Timeout:            S5TIME:= S5T#700MS;
    MCP1Cycl:               S5TIME:= S5T#200MS;
    MCP2In:                 POINTER;                    //Start addr. input signals MCP 2
    MCP2Out:                POINTER;                    //Start addr. output signals MCP 2
    MCP2StatSend:           POINTER;                    //Status DW for sending MCP 2
    MCP2StatRec:            POINTER;                    //Status DW for receiving MCP 2
    MCP2BusAdr:             INT;
    MCP2Timeout:            S5TIME:= S5T#700MS;
    MCP2Cycl:               S5TIME:= S5T#200MS;
    MCPMPI:                 BOOL:= FALSE;
    MCP1Stop:               BOOL:= FALSE;
    MCP2Stop:               BOOL:= FALSE;
    MCP1NotSend:            BOOL:= FALSE;
    MCP2NotSend:            BOOL:= FALSE;
    MCPSDB210:              BOOL:= FALSE;
    MCPCopyDB77:            BOOL:= FALSE;
    MCPBusType:             BYTE=B#16#0;
    HHU:                    INT:=0;                     //Handheld unit interface
                                                        //0: No HHU
                                                        //1: HHU on MPI
                                                        //2: HHU on OPI
    BHGIn:                  POINTER;                    //Transmit data of the HHU
    BHGOut:                 POINTER;                    //Receive data of the HHU
    BHGStatSend:            POINTER;                    //Status DW for sending HHU
    BHGStatRec:             POINTER;                    //Status DW for receiving HHU
    BHGInLen:               BYTE:= B#16#6;              //Input 6 bytes
    BHGOutLen:              BYTE:= B#16#14;             //Output 20 bytes
    BHGTimeout:             S5TIME:= S5T#700MS;
    BHGCycl:                S5TIME:= S5T#100MS;
    BHGRecGDNo:             INT:=2;
    BHGRecGBZNo:            INT:=2;
    BHGRecObjNo:            INT:=1;
    BHGSendGDNo:            INT:=2;
```

```
    BHGSendGBZNo:           INT:=1;
    BHGSendObjNo:           INT:=1;
    BHGMPI:                 BOOL:= FALSE;
    BHGStop:                BOOL:= FALSE;
    BHGNotSend:             BOOL:= FALSE;
    NCCyclTimeout:          S5TIME:= S5T#200MS;
    NCRunupTimeout:         S5TIME:= S5T#50S;
    ListMDecGrp:            INT:=0;
    NCKomm:                 BOOL:= FALSE;
    MMCToIF:                BOOL:=TRUE;
    HWheelMMC:              BOOL:=TRUE;             //Handwheel selection via HMI
    ExtendAlMsg :           BOOL;
    MsgUser:                INT:=10;                //Number of user areas in DB 2
    UserIR:                 BOOL:= FALSE;           //User programs in OB 40,
                                                    //Observe local data expansion!
    IRAuxfuT:               BOOL:= FALSE;           //Evaluate T function in OB 40
    IRAuxfuH:               BOOL:= FALSE;           //Evaluate H function in OB 40
    IRAuxfuE:               BOOL:= FALSE;           //Evaluate DL function in OB 40
    UserVersion:            POINTER;                //Pointer to string variable indicated in
                                                    //version screen display
    OpKeyNum :              INT;
    Op1KeyIn                POINTER;
    Op1KeyOut :             POINTER;
    Op1KeyBusAdr :          INT;
    Op2KeyIn :              POINTER;
    Op2KeyOut :             POINTER;
    Op2KeyBusAdr :          INT;
    Op1KeyStop :            BOOL;
    Op2KeyStop :            BOOL;
    Op1KeyNotSend :         BOOL;
    Op2KeyNotSend :         BOOL;
    OpKeyBusType :          BYTE ;
    IdentMcpBusAdr :        INT;
    IdentMcpProfilNo :      BYTE ;
    IdentMcpBusType :       BYTE ;
    IdentMcpStrobe :        BOOL;
END_VAR
```

```
VAR_OUTPUT
    MaxBAG:                 INT;
    MaxChan:                INT;
    MaxAxis:                INT;
    ActivChan:              ARRAY[1..10] OF BOOL;
    ActivAxis:              ARRAY[1..31] OF BOOL;
    UDInt :                 INT;
    UDHex:                  INT;
    UDReal :                INT;
    IdentMcpType :          BYTE ;
    IdentMcpLengthIn :      BYTE ;
    IdentMcpLengthOut :     BYTE ;
END_VAR
```

Description of formal parameters of SINUMERIK 840D sl

The table below lists all formal parameters of the RUN_UP function for the 840D sl:

Signal	Type	Type	Range of values	Remark
MCPNum	I	INT	Up to 2	Number of active MCP 0: No MCPs available
MCP1In MCP2In	I	POINTER	E0.0 to E120.0 or M0.0 to M248.0 or DBn DBX0.0 to DBXm.0	Start address for input signals of relevant machine control panel
MCP1Out MCP2Out	I	POINTER	A0.0 to A120.0 or M0.0 to M248.0 or DBn DBX0.0 to DBXm.0	Start address for output signals of relevant machine control panel
MCP1StatSend MCP2StatSend	I	POINTER	A0.0 to A124.0 or M0.0 to M252.0 or DBn DBX0.0 to DBXm.0	Currently no significance
MCP1StatRec MCP2StatRec	I	POINTER	A0.0 to A124.0 or M0.0 to M252.0 or DBn DBX0.0 to DBXm.0	Currently no significance
MCP1BusAdr MCP2BusAdr	I	INT	1 ... 126 192 .. 223	DP slave: PROFIBUS address Ethernet MCP: DIP-Setting

Signal	Type	Type	Range of values	Remark
MCP1Timeout MCP2Timeout	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for machine control panel
MCP1Cycl MCP2Cycl	I	S5time	Recommendation: 200 ms	Relevant only for PROFIBUS
MCPMPI	I	BOOL	False	Available owing to compatibility
MCP1Stop MCP2Stop	I	BOOL		0: Start transfer of machine control panel signals 1: Stop transfer of machine control panel signals DP slave: Slave deactivated
MCP1NotSend MCP2NotSend	I	BOOL		0: Send and receive operation activated 1: Receive machine control panel signals only
MCPSDB210	I	BOOL	False	Available owing to compatibility
MCPCopyDB77	I	BOOL	False	Available owing to compatibility
MCPBusType	I	BYTE		Righthand half byte (bits 0...3) for MCP1 Lefthand half byte (bits 4...7) for MCP2 b#16#33: PROFIBUS b#16#44: PROFIBUS on the MPI/DP port b#16#55: Ethernet B#16#66: PROFINET Mixed mode possible, see Section "Configuration machine control panel, handheld unit, direct keys (Page 962)"
HHU	I	INT	0, 5	Handheld unit interface 0: No HHU 5: HHU on Ethernet
BHGIn	I	POINTER	E0.0 to E124.0 or M0.0 to M252.0 or DBn DBX0.0 to DBXm.0	Start address PLC receive data from HHU
BHGOut	I	POINTER	A0.0 to A124.0 or M0.0 to M252.0 or DBn DBX0.0 to DBXm.0	Start address PLC transmit data to HHU

Signal	Type	Type	Range of values	Remark	
BHGStatSend	I	POINTER	A0.0 to A124.0 or M0.0 to M252.0 or DBn DBX0.0 to DBXm.0	Available owing to compatibility	
BHGStatRec	I	POINTER	A0.0 to A124.0 or M0.0 to M252.0 or DBn DBX0.0 to DBXm.0	Available owing to compatibility	
BHGInLen	I	BYTE	HHU default: B#16#6 (6 Byte)	Available owing to compatibility	
BHGOutLen	I	BYTE	HHU default: B#16#14 (20 Byte)	Available owing to compatibility	
BHGTimeout	I	S5time	Recommendation: 700 ms	Available owing to compatibility	
BHGCycl	I	S5time	Recommendation: 100 ms	Available owing to compatibility	
BHGRecGDNo	I	INT	HHU default: 2	Ethernet DIP switch	
BHGRecGBZNo	I	INT	HHU default: 2	Available owing to compatibility	
BHGRecObjNo	I	INT	HHU default: 1	Available owing to compatibility	
BHGSendGDNo	I	INT	HHU default: 2	Available owing to compatibility	
BHGSendGBZNo	I	Int	HHU default: 1	Available owing to compatibility	
BHGSendObjNo	I	INT	HHU default: 1	Available owing to compatibility	
BHGMPI	I	BOOL	False	Available owing to compatibility	
BHGStop	I	BOOL		0:	Start transmission of handheld unit signals
				1:	Stop transmission of handheld unit signals
BHGNotSend	I	BOOL		0:	Send and receive operation activated
				1:	Receive handheld unit signals only
NCCyclTimeout	I	S5time	Recommendation: 200 ms	Cyclic sign-of-life monitoring NCK	
NCRunupTimeout	I	S5time	Recommendation: 50 s	Power-up monitoring NCK	
ListMDecGrp	I	INT	0 ... 16	Activation of expanded M group decoding	
				0:	Not active
				1...16:	Number of M groups
NCKomm	I	BOOL		PLC NC communications services (FB 2/3/4/5/7: Put/Get/PI_SERV/GETGUD)	
				TRUE:	Active

Signal	Type	Type	Range of values	Remark	
MMCToIF	I	BOOL		Transmission of HMI signals to interface (modes, program control etc.)	
				TRUE:	Active
HWheelMMC	I	BOOL		TRUE:	Handwheel selection via HMI
				FALSE:	Handwheel selection via user program
ExtendAlMsg	I	BOOL		Activation extension of the FC10 (see Section "Structure and functions of the basic program (Page 941)")	
MsgUser	I	INT	0 ... 32	Number of user areas for messages (DB 2)	
UserIR	I	BOOL		Local data expansion OB40 required for processing of signals from user	
IRAuxfuT	I	BOOL		Evaluate T function in OB 40	
IRAuxfuH	I	BOOL		Evaluate H function in OB 40	
IRAuxfuE	I	BOOL		Evaluate DL function in OB 40	
UserVersion	I	POINTER	DBxx	Pointer to string variable. The associated string variable is indicated in the version display (max. 41 characters).	
OpKeyNum	I	INT	0 ... 2	Number of active Direct control key modules	
				0:	no Ethernet direct control keys available.
Op1KeyIn Op2KeyIn	I	POINTER	P#Ex.0 or P#Mx.0 or P#DBn.DBXx.0.	Start address for the input signals of the affected direct control key modules	
Op1KeyOut Op2KeyOut	I	POINTER	P#Ax.0 or P#Mx.0 or P#DBn.DBXx.0.	Start address for the output signals of the affected direct control key modules	
Op1KeyBusAdr Op2KeyBusAdr	I	INT	1 ... 191	Direct control keys via Ethernet: TCU index:	
Op1KeyStop Op2KeyStop	I	BOOL		0:	Start transmission of direct control key signals
				1:	Stop transmission of direct control key signals
Op1KeyNotSend Op2KeyNotSend	I	BOOL		0:	Send and receive operation activated
				1:	Receive direct control key signals only
OpKeyBusType	I	BYTE	b#16#55	b#16#55: Ethernet	
IdentMcpBusAdr	I	INT	1 ... 254	only IE devices	
IdentMcpProfilNo	I	BYTE	0, 1	Profile of a device	
				0:	complete device

Signal	Type	Type	Range of values	Remark
				1: only direct control keys
IdentMcpBusType	I	BYTE	b#16#5	only IE devices
IdentMcpStrobe	I	BOOL		Activate query
MaxBAG	O	INT	1 ... 10	Number of mode groups
MaxChan	O	INT	1 ... 10	Number of channels
MaxAxis	O	INT	1 ... 31	Number of axes
ActivChan	O	ARRAY[1...10] OF BOOL		Bit string for active channels
ActivAxis	O	ARRAY [1..31] OF BOOL		Bit string for active axes
UDInt	O	INT		Quantity of INTEGER machine data in DB20
UDHex	O	INT		Quantity of hexadecimal machine data in DB20
UDReal	O	INT		Quantity of REAL machine data in DB 20
IdentMcpType	O	BYTE		Type (HT2, HT8, ...)
IdentMcpLengthIn	O	BYTE		Length info input data in PLC
IdentMcpLengthOut	O	BYTE		Length info output data in PLC

MCP/HHU monitoring (840D sl)

The following alarms are displayed at HMI in cases of errors for the communication with the machine control panel (MCP):

- 400260: MCP 1 failure or
- 400261: MCP 2 failure
- 400262: HHU failure

In this case, the input signals from the MCP or from the handheld unit (MCP1In/MCP2In or BHGIn) are reset to 0. If it is possible to resynchronize the PLC and MCP/HHU, communication is resumed automatically and the error message reset by the GP.

Call example for 840D sl

An example call for the FB 1 in OB 100 appears below. This example is part of the diskette with basic program for 840D sl.

```
ORGANIZATION_BLOCK OB 100
VAR_TEMP
    OB100_EV_CLASS :                BYTE ;
    OB100_STRTUP :                  BYTE ;
    OB100_PRIORITY :                BYTE ;
    OB100_OB_NUMBR :                BYTE ;
    OB100_RESERVED_1 :              BYTE ;
    OB100_RESERVED_2 :              BYTE ;
    OB100_STOP :                    WORD ;
    OB100_RESERVED_3 :              WORD ;
    OB100_RESERVED_4 :              WORD ;
    OB100_DATE_TIME :               DATE_AND_TIME;
END_VAR
BEGIN
    CALL FB 1, DB 7 (
          MCPNum :=                 1,
          MCP1In :=                 P#E0.0,
          MCP1Out :=                P#A0.0,
          MCP1StatSend :=           P#A8.0,
          MCP1StatRec :=            P#A12.0,
          MCP1BusAdr :=             6,
          MCP1Timeout :=            S5T#700MS,
          MCP1Cycl :=               S5T#200MS,
          NC-CyclTimeout :=         S5T#200MS,
          NC-RunupTimeout :=        S5T#50S);
//INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

12.15.2 FB 2: Read GET NC variable

Function

The FB 2 "GET" function block can be used to read variables from the area of the NC from the PLC user program. The FB is multi-instance-capable.

FB 2 also includes an Instance DB from the user area.

When FB 2 is called with a positive signal edge change at control input "Req", a job is started, which reads the NCK variables referenced by "Addr1" to "Addr8" and then copies them to the PLC operand areas referenced by "RD1" to "RD8". Successful completion of the read process is indicated by a logical "1" in status parameter "NDR".

The **read operation** lasts for several PLC cycles (normal case: 1 - 2 PLC cycles). The block can be called up in cyclic mode only.

Any errors are displayed via "Error" and "State".

In order to reference the NC variables, they are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this data block in the signal list. When calling FB 2, the variable addresses are then transferred in the following form for parameters "Addr1" to "Addr8":

"<DB name>.<S7 name>".

Variable addressing

For some NC variables, it is necessary to select "Area no." and/or "Line" or "Column" from the NC VAR selector. For these variables it is possible to select a basic type, i.e. "Area no.", "Line" and "Column" are preassigned "0".

The contents of the "Area no.", "Line" and "Column" specified by the NC VAR selector are checked for a "0" in FB 2. If a "0" is present, the value is transferred to the input parameter.

Before calling FB 2, the user must supply the appropriate parameters:

Parameter: FB 2	Parameter: NC VAR selector
"Unit<x>"	"Area no."
"Column<x>"	"Column"
"Line<x>"	"Line"
where <x> = 1 - 8	

NOTICE

FB 2 can read NC variables only if basic program parameter NCKomm "1" has been set to "1" (in OB 100: FB 1, DB 7). The call is permitted only in cyclic program OB1. An assignment for all parameters with "Req" = 0 is also permitted in OB 100.

Channel-specific variables
When channel-specific variables are read, only variables from **one** channel may be addressed via "Addr1" to "Addr8" when FB 2 is called.

Drive-specific variables
When drive-specific variables are read, only variables from **one** SERVO drive object may be addressed via "Addr1" to "Addr8" when FB 2 is called. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.

Error case
In the event of an error, reading of variables from different channels or drive objects, or simultaneously from a channel and a drive object, the following is provided as feedback:

- "Error" == TRUE
- "State" == W#16#02

Variables within **one** group can be combined in a job:

	Area				
Group 1	C[1]	N	B	A	T
Group 2	C[2]	N	B	A	T
Group 3	V[.]	H[.]			
The same rules apply for channels 3 to 10 as for group 1 and group 2 shown in the example.					

Note

The number of usable variables can be less than eight when simultaneously reading several variables of the "String" type.

Declaration of the function

```
FUNCTION_BLOCK FB 2
VAR_INPUT
    Req :                   BOOL;
    NumVar :                INT;
    Addr1 :                 ANY ;
    Unit1 :                 BYTE ;
    Column1 :               WORD ;
    Line1 :                 WORD ;
    Addr2 :                 ANY ;
    Unit2 :                 BYTE ;
    Column2 :               WORD ;
    Line2 :                 WORD ;
    Addr3 :                 ANY ;
    Unit3 :                 BYTE ;
    Column3 :               WORD ;
    Line3 :                 WORD ;
    Addr4 :                 ANY ;
    Unit4 :                 BYTE ;
    Column4 :               WORD ;
    Line4 :                 WORD ;
    Addr5 :                 ANY ;
    Unit5 :                 BYTE ;
    Column5 :               WORD ;
    Line5 :                 WORD ;
    Addr6 :                 ANY ;
    Unit6 :                 BYTE ;
    Column6 :               WORD ;
    Line6 :                 WORD ;
    Addr7 :                 ANY ;
    Unit7 :                 BYTE ;
    Column7 :               WORD ;
    Line7 :                 WORD ;
    Addr8 :                 ANY ;
    Unit8 :                 BYTE ;
    Column8 :               WORD ;
    Line8 :                 WORD ;
END_VAR
VAR_OUTPUT
    Error :                 BOOL;
    NDR :                   BOOL;
    State :                 WORD ;
END_VAR
```

```
VAR_IN_OUT
    RD1 :                   ANY ;
    RD2 :                   ANY ;
    RD3 :                   ANY ;
    RD4 :                   ANY ;
    RD5 :                   ANY ;
    RD6 :                   ANY ;
    RD7 :                   ANY ;
    RD8 :                   ANY ;
END_VAR
```

Description of formal parameters

The following table shows all formal parameters of FB 2.

Parameter	Type	Type	Range of values	Remark
Req	I	BOOL	-	Job start with positive signal edge
NumVar	I	INT	1 ... 8	Number of variables to be read: Addr1 - Addr8
Addr1 - Addr8	I	ANY	[DBName].[VarName]	Variable identifiers from **NC Var selector**
Unit1 - Unit8	I	BYTE	-	Area address, optional for variable addressing
Column1 - Column8	I	WORD	-	Column address, optional for variable addressing
Line1 - Line8	I	WORD	-	Line address, optional for variable addressing
Error	O	BOOL	-	Negative acknowledgement of job or execution of job impossible
NDR	O	BOOL	-	Job successfully executed Data is available
State	O	WORD	-	See error identifiers
RD1 - RD8	I/O	ANY	P#Mm.n BYTE x... P#DBnr.dbxm.n BYTE x	Target area for read data

Error identifiers

If it was not possible to execute a job, the failure is indicated by "logic 1" on status parameter "Error". The error cause is coded at the block output State:

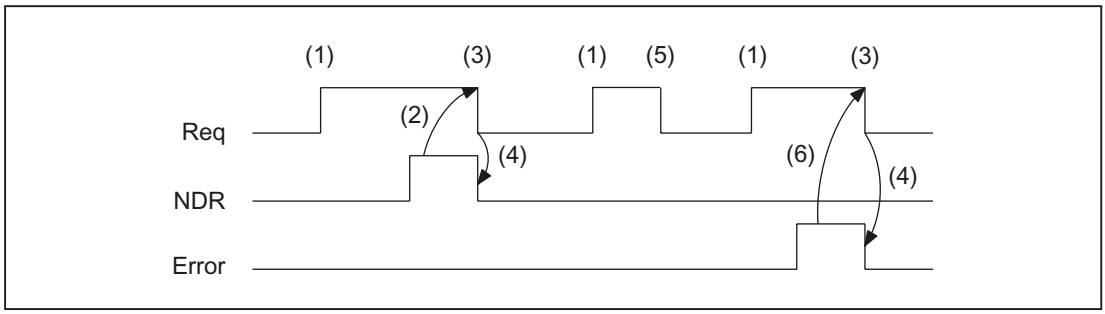
State		Meaning	Note
WORD H	WORD L		
1 - 8	1	Access error	WORD H: Number of the variable in which the error occurred
0	2	Error in job	Incorrect compilation of variables in a job
0	3	Negative acknowledgement, job not executable	Internal error, try: • Check job • NC reset
1 - 8	4	Insufficient local user memory available	Read variable is longer than specified in "RD1" - "RD8"; WORD H: Number of the variable in which the error occurred
0	5	Format conversion error	Error on conversion of var. type double: Variable is not within the S7 REAL area
0	6	FIFO full	Job must be repeated since queue is full
0	7	Option not set	BP parameter "NCKomm" is not set
1 - 8	8	Incorrect target area (RD)	RD1 to RD8 may not be local data
0	9	Transmission occupied	Job must be repeated
1 - 8	10	Error in variable addressing	"Unit" or "Column"/"Line" contains value 0
0	11	Address of variable invalid	Check "Addr" (or variable name), "Area", "Unit"
0	12	NumVar = 0	Check parameter NumVar
1 - 8	13 (0x0d)	ANY data reference incorrect	NcVar data required has not been parameterized

Configuration steps

Proceed as follows to read NC variables:

- Select variables with the NC VAR selector.
- Save selected variables in a *.VAR file.
- Generate a STEP 7 *.STL source file.
- Generate a DB with the associated address data.
- Enter the symbol for the generated DB in the symbol table so that it is possible to access the address parameters symbolically in the user program.
- Parameterization of FB 2.

Pulse diagram



(1) Activation of function
(2) Positive acknowledgement: Receive new data
(3) Reset function activation after receipt of acknowledgement
(4) Signal change by means of FB
(5) Not permissible
(6) Negative acknowledgement: Error has occurred, error code in the output parameter State

Call example

Reading of three channel-specific machine data from channel 1, whose address specifications are stored in DB120.

Select data with NC VAR selector and store in file DB120.VAR; then create file DB120.AWL:

Area	Block	Name	Type	No.	Byte	S7 Name
C[1]	M	MD20070 $MC_AXCONF_MACHAX_USED[1]	CHAR	20070	1	C1AxConfMachAx Used1
C[1]	M	MD20070 $MC_AXCONF_MACHAX_USED[2]	CHAR	20070	1	C1AxConfMachAx Used2
C[1]	M	MD20090 $MC_SPIND_DEF_MASTER_SPIND	INT	20090	1	C1SpindDefMaster Spind

S7 (ALIAS) names have been selected in order to:

- Incorporate the channel designation into the name
  and
- Remove the characters [ ] which are not legal in a STEP 7 symbol.

Entry of the name in the S7 SYMBOL table (e.g. NCVAR for DB120):

Symbol	Operand	Data type
NCVAR	DB 120	DB 120

File DB120.AWL must be compiled and transferred to the PLC.

Parameterization of FB 2 with instance DB 110:

```
DATA_BLOCK DB 110           //Unassigned user DB, as instance for FB 2
FB 2
BEGIN
END_DATA_BLOCK
Function FC "VariablenCall" : VOID
   U       I 7.7;                          //Unassigned machine control panel key
   S       M 100.0;                        //Activate req.
   U       M 100.1;                        //NDR completed message
   R       M 100.0;                        //Terminate job
   U       I 7.6;                          //Manual error acknowledgement
   U       M 102.0;                        //Error pending
   R       M 100.0;                        //Terminate job
   CALL FB 2, DB 110 (
      Req :              M 100.0,
      NumVar :           3,              //Read three variables
      Addr1 :            NCVAR.C1AxConfMachAxUsed1,
      Addr2 :            NCVAR.C1AxConfMachAxUsed2,
      Addr3 :            NCVAR.C1SpindDefMasterSpind,
      Error :            M102.0,
      NDR :              M100.1,
      State :            MW104,
      RD1 :              P#DB99.DBX0.0 BYTE 1,
      RD2 :              P#DB99.DBX1.0 BYTE 1,
      RD3 :              P#M110.0 INT 1);
```

Example: Variable addressing

Reading of two R parameters from channel 1, whose address specifications are stored in DB 120 as the basic type. The R parameter number is parameterized via parameter Line<x>.

```
DATA_BLOCK DB 120
VERSION : 0.0
STRUCT
   C1_RP_rpa0_0:
   STRUCT
   SYNTAX_ID :                  BYTE := B#16#82;
   area_and_unit :              BYTE := B#16#41;
   column :                     WORD := W#16#1;
   line :                       WORD := W#16#0;
   block type :                 BYTE := B#16#15;
   NO. OF LINES :               BYTE := B#16#1;
   type :                       BYTE := B#16#F;
   length :                     BYTE := B#16#8;
   END_STRUCT;
END_STRUCT;
BEGIN
END_DATA_BLOCK
CALL FB 2, DB 110 (
   Req :                                M 0.0,
   NumVar :                             2,
   Addr1 :                              "NCVAR".C1_RP_rpa0_0,
   Line1 :                              W#16#1,
   Addr2 :                              "NCVAR".C1_RP_rpa0_0,
   Line2 :                              W#16#2,
   Error :                              M 1.0,
   NDR :                                M 1.1,
   State :                              MW 2,
   RD1 :                                P#M 4.0 REAL 1,
   RD2 :                                P#M 24.0 REAL 1);
```

Data types

The data types of the NCK are listed in the NC VAR selector with the variables. The tables below give the assignments to the S7 data types.

Classification of data types	
NCK data type	**S7 data type**
double	REAL
double	REAL2
float	REAL
long	DINT
integer	DINT
uint_32	DWORD
int_16	INT
uint_16	WORD
unsigned	WORD
char	CHAR or BYTE
string	STRING
bool	BOOL
datetime	DATE_AND_TIME

In order to read a variable of the "double" type from the NCK without adapting the format, an ANY pointer of the REAL 2 type must be specified in the target area for read data (e.g. P#M100.0 REAL 2). If the basic program recognizes REAL 2 as the target type when reading a variable of the "double" type, the data is applied to the PLC data area as a 64-bit floating-point number.

12.15.3 FB 3: PUT write NC variables

Function

The FB 3 "PUT" function block can be used to write variables to the area of the NC from the PLC user program. The FB is multi-instance-capable.

Every FB 3 call must be assigned a separate instance DB from the user area.

When FB 3 is called with a positive signal edge change at control input "Req", a job is started to overwrite the NC variables referenced by "Addr1" to "Addr8" with the data of the PLC operand areas locally referenced by "SD1" to "SD8". Successful completion of the write process is indicated by a logical "1" in status parameter "Done".

The **write operation** lasts for several PLC cycles (normal case: 1 - 2 PLC cycles). The block can be called up in cyclic mode only.

Any errors are displayed via "Error" and "State".
In order to reference the NC variables, all required variables are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this DB in the symbol table. When calling FB 3, the variable addresses are then transferred in the following form for parameters "Addr1" to "Addr8":

"<DB name>.<S7 name>".

Variable addressing

For some NC variables, it is necessary to select "Area no." and/or "Line" or "Column" from the NC VAR selector. For these variables it is possible to select a basic type, i.e. "Area no.", "Line" and "Column" are preassigned "0".

The contents of the "Area no.", "Line" and "Column" specified by the NC VAR selector are checked for a "0" in FB 3. If a "0" is present, the value is transferred to the input parameter.

Before calling FB 3, the user must supply the appropriate parameters:

Parameter: FB 3	Parameter: NC VAR selector
"Unit<x>"	"Area no."
"Column<x>"	"Column"
"Line<x>"	"Line"
where <x> = 1 - 8	

Machine data, GUD

In order to define machine data and GUD without a password, the protection level of the data you want to access must be redefined to the lowest level.

References:

- Commissioning Manual; Section: "Protection levels concept"
- Programming Manual, Job Planning; Section: "Define protection levels for user data"

NOTICE

FB 3 can only write NC variables if basic program parameter "NCKomm" has been set to "1" (in OB 100: FB 1, DB 7). The call is permitted only in cyclic program OB1. An assignment for all parameters with "Req" = 0 is also permitted in OB 100.

Channel-specific variables
When channel-specific variables are written, only variables from **one** channel may be addressed via "Addr1" to "Addr8" when FB 3 is called.

Drive-specific variables
When drive-specific variables are written, only variables from **one** SERVO drive object may be addressed via "Addr1" to "Addr8" when FB 3 is called. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.

Error case
In the event of an error, writing of variables from different channels or drive objects, or simultaneously from a channel and a drive object, the following is provided as feedback:

- "Error" == TRUE
- "State" == W#16#02

NCK variables within **one** group can be combined in a job:

	Area				
Group 1	C[1]	N	B	A	T
Group 2	C[2]	N	B	A	T
Group 3	V[.]	H[.]			
The same rules apply for channels 3 to 10 as for group 1 and group 2 shown in the example.					

Note

The number of usable variables can be less than eight when simultaneously writing several variables of the "String" type.

Declaration of the function

```
FUNCTION_BLOCK FB 3
VAR_INPUT
    Req :                BOOL;
    NumVar :             INT;
    Addr1 :              ANY ;
    Unit1 :              BYTE ;
    Column1 :            WORD ;
    Line1 :              WORD ;
    Addr2 :              ANY ;
    Unit2 :              BYTE ;
    Column2 :            WORD ;
    Line2 :              WORD ;
    Addr3 :              ANY ;
    Unit3 :              BYTE ;
    Column3 :            WORD ;
    Line3 :              WORD ;
    Addr4 :              ANY ;
    Unit4 :              BYTE ;
    Column4 :            WORD ;
    Line4 :              WORD ;
    Addr5 :              ANY ;
    Unit5 :              BYTE ;
    Column5 :            WORD ;
    Line5 :              WORD ;
    Addr6 :              ANY ;
    Unit6 :              BYTE ;
    Column6 :            WORD ;
    Line6 :              WORD ;
    Addr7 :              ANY ;
    Unit7 :              BYTE ;
    Column7 :            WORD ;
    Line7 :              WORD ;
    Addr8 :              ANY ;
    Unit8 :              BYTE ;
    Column8 :            WORD ;
    Line8 :              WORD ;
END_VAR
VAR_OUTPUT
    Error :              BOOL;
    Done :               BOOL;
    State :              WORD ;
END_VAR
```

```
VAR_IN_OUT
     SD1 :                    ANY ;
     SD2 :                    ANY ;
     SD3 :                    ANY ;
     SD4 :                    ANY ;
     SD5 :                    ANY ;
     SD6 :                    ANY ;
     SD7 :                    ANY ;
     SD8 :                    ANY ;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the PUT function.

Signal	Type	Type	Range of values	Remark
Req	I	BOOL	-	Job start with positive signal edge
NumVar	I	INT	1 ... 8	Number of variables to be written: Addr1 - Addr8
Addr1 - Addr8	I	ANY	[DBName].[VarName]	Variable identifiers from **NC Var selector**
Unit 1 - Unit 8	I	BYTE	-	Area address, optional for variable addressing
Column 1 - Column 8	I	WORD	-	Column address, optional for variable addressing
Line 1 - Line 8	I	WORD	-	Line address, optional for variable addressing
Error	O	BOOL	-	Negative acknowledgement of job or execution of job impossible
Done	O	BOOL	-	Job successfully executed
State	O	WORD	-	See error identifiers
SD1 - SD8	I/O	ANY	P#Mm.n BYTE x... P#DBnr.dbxm.n BYTE x	Data to be written

Error identifiers

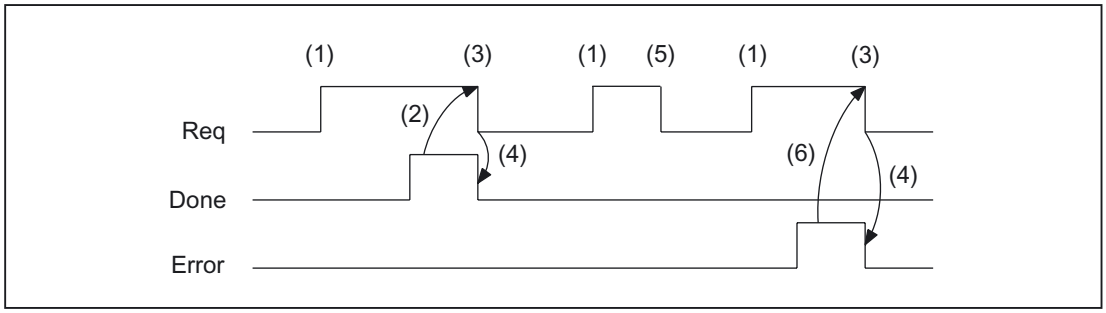
If it was not possible to execute a job, the failure is indicated by "logic 1" on status parameter error. The error cause is coded at the block output State:

State		Meaning	Note
WORD H	WORD L		
1 - 8	1	Access error	WORD H: Number of the variable in which the error occurred
0	2	Error in job	Incorrect compilation of variables in a job
0	3	Negative acknowledgement, job not executable	Internal error, try: • Check job • NC reset
1 - 8	4	Data areas or data types do not match or string is empty	Check data to be written in "SD1" - "SD8"; WORD H: Number of the variable in which the error occurred
0	6	FIFO full	Job must be repeated since queue is full
0	7	Option not set	BP parameter "NCKomm" is not set
1 - 8	8	Incorrect target area (SD)	"SD1" - "SD8" must not be local data
0	9	Transmission occupied	Job must be repeated
1 - 8	10	Error in variable addressing	"Unit" or "Column"/"Line" contains value 0
0	11	Variable address invalid or variable is read-only	Check "Addr" (or variable name), "Area", "Unit"
0	12	NumVar = 0	Check parameter NumVar
1 - 8	13 (0x0d)	ANY data reference incorrect	NcVar data required has not been parameterized
1 - 8	15 (0x0f)	User data too long	Remedy: Write fewer variables per job or use shorter string variables

Configuration steps

To write NC variables, the same configuration steps are required as for reading NC variables. It is useful to store the address data of all NC variables to be read or written in a DB.

Pulse diagram



(1) Activation of function
(2) Positive acknowledgement: variables have been written
(3) Reset function activation after receipt of acknowledgement
(4) Signal change by means of FB
(5) Not permissible
(6) Negative acknowledgement: Error has occurred, error code in output parameter state

Call example

Writing of three channel-specific machine data of channel 1:

Select the three data with NC VAR selector and store in the file DB120.VAR:

Area	Block	Name	Type	Byte	S7 Name
C[1]	RP	rpa[5]	DOUBLE	4	rpa_5C1RP
C[1]	RP	rpa[11]	DOUBLE	4	rpa_11C1RP
C[1]	RP	rpa[14)	DOUBLE	4	rpa_14C1RP

Entry NCVAR for DB 120 with the S7 SYMBOL Editor:

Symbol	Operand	Data type
NCVAR	DB 120	DB 120

File DB120.AWL must be compiled and transferred to the PLC.

Call and parameterization of FB 3 with instance DB 111:

```
DATA_BLOCK DB 111          //Unassigned user DB, as instance for FB 3
FB 3
BEGIN
Function FC "VariablenCall" : VOID
END_DATA_BLOCK
   U       I 7.7;          //Unassigned machine control panel key
   S       M 100.0;        //Activate req.
   U       M 100.1;        //Done completed message
   R       M 100.0;        //Terminate job
   U       I 7.6;          //Manual error acknowledgement
   U       M 102.0;        //Error pending
   R       M 100.0;        //Terminate job
   CALL FB 3, DB 111 (
       Req :              M 100.0,
       NumVar :           3,                          //Write three variables
       Addr1 :            NCVAR.rpa_5C1RP,
       Addr2 :            NCVAR.rpa_11C1RP,
       Addr3 :            NCVAR.rpa_14C1RP,
       Error :            M102.0,
       Done :             M100.1,
       State :            MW104,
       SD1 :              P#DB99.DBX0.0 REAL 1,
       SD2 :              P#DB99.DBX4.0 REAL 1,
       SD3 :              P#M110.0 REAL 1);
```

Example: Variable addressing

Writing of two R parameters of channel 1, whose address specifications are stored in DB 120 as the basic type. The R parameter number is parameterized via parameter LineX.

```
DATA_BLOCK DB 120
VERSION : 0.0
STRUCT
   C1_RP_rpa0_0:
   STRUCT
   SYNTAX_ID :                BYTE := B#16#82;
   area_and_unit :            BYTE := B#16#41;
   column :                   WORD := W#16#1;
   line :                     WORD := W#16#0;
   block type :               BYTE := B#16#15;
   NO. OF LINES :             BYTE := B#16#1;
   type :                     BYTE := B#16#F;
   length :                   BYTE := B#16#8;
   END_STRUCT;
END_STRUCT;
BEGIN
END_DATA_BLOCK
CALL FB 3, DB 122 (
   Req :                      M 10.0,
   NumVar :                   2,
   Addr1 :                    "NCVAR".C1_RP_rpa0_0,
   Line1 :                    W#16#1,
   Addr2 :                    "NCVAR".C1_RP_rpa0_0,
   Line3 :                    W#16#2
   Error :                    M 11.0,
   Done :                     M 11.1,
   State :                    MW 12,
   SD1 :                      P#M 4.0 REAL 1,
   SD2 :                      P#M 24.0 REAL 1);
```

12.15.4 PI services

12.15.4.1 FB 4: PI_SERV PI service request

Function

The function block FB 4 "PI_SERV" can be used to start program instance services (PI services) in the NC area. Every FB 4 call must be assigned an instance DB from the user area.

Note

It is recommended that instead of FB 4 the extended function block FB 7 is used. See Section "FB 7: PI_SERV2 (PI service request) (Page 1065)".

The required PI service must be referenced via the "PIService" parameter. The Addr and WVar parameters can be used to parameterize the selected PI service.

The job is started when FB 4 is called by means of a positive edge change at control input "Req". Successful execution of the job is displayed by means of a logical "1" at the "Done" output. Any errors are displayed via the "Error" and "State" outputs.

The DB 16 "PI" data block contains internal descriptions of the possible PI services. A name must then be assigned to this data block in the signal list. On calling the FB 4, "DB-Name.PI-Name" is transferred as the actual parameter for "PIService".

The execution of the PI service generally extends over several PLC cycles.

Requirements

FB 4 can start PI services only if the basic program parameter NCKomm = 1 has been set (OB 100: FB 1, DB 7).

Supplementary conditions

The call is permitted only in cyclic program OB1. Writing of the parameters in OB 100 is also permissible without starting the request (Req = 0).

Declaration of the function

```
FUNCTION_BLOCK FB 4
VAR_INPUT
w   Req :                   BOOL;
    PIService :             ANY ;
    Unit :                  INT;
    Addr1 :                 ANY ;
    Addr2 :                 ANY ;
    Addr3 :                 ANY ;
    Addr4 :                 ANY ;
    WVar1 :                 WORD ;
    WVar2 :                 WORD ;
    WVar3 :                 WORD ;
    WVar4 :                 WORD ;
    WVar5 :                 WORD ;
    WVar6 :                 WORD ;
    WVar7 :                 WORD ;
    WVar8 :                 WORD ;
    WVar9 :                 WORD ;
    WVar10 :                WORD ;
END_VAR
VAR_OUTPUT
    Error :                 BOOL;
    Done :                  BOOL;
    State :                 WORD ;
END_VAR
```

Description of formal parameters

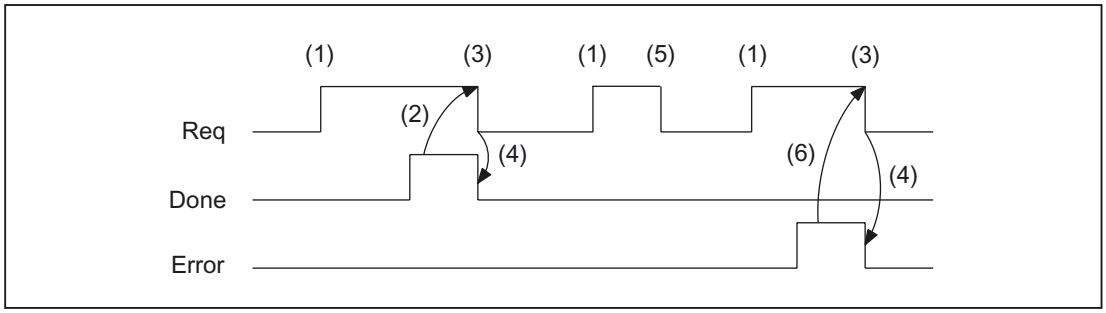
Signal	Type	Type	Range of values	Remark
Req	I	BOOL		Job request
PIService	I	ANY	[DBName].[VarName] standard: "PI".[VarName]	Designation of the PI service, see Section "List of available PI services (Page 1028)"
Unit	I	INT	1...	Area number
Addr1 to Addr4	I	ANY	[DBName].[VarName]	Reference to strings Specification according to selected PI service,
WVar1 to WVar10	I	WORD	1...	INTEGER or WORD variables Specification according to selected PI service,
Error	O	BOOL	TRUE/FALSE	Negative acknowledgement of job or execution of job impossible
Done	O	BOOL	TRUE/FALSE	Job successfully executed
State	O	WORD	See error identifiers	-
I: Input O: Output				

Error identifiers

If it was not possible to execute a job, the failure is indicated by "logic 1" at "Error" output. The cause of the error is displayed at the "State" output:

State	Meaning	Note
3	Negative acknowledgement, job not executable	Internal error, possible remedy through an NC RESET
6	FIFO full	Repeat the job, queue is full
7	Option not set	FB 1, parameter "NCKomm" is not set
9	Transmission occupied	Repeat the command
13 (0x0d)	Addr1.. Adddr4: Reference invalid	Specify missing string
14 (0x0e)	"PIService": Reference unknown	No valid PI designation
15 (0x0f)	Addr1.. Adddr4: String too long	Check string lengths

Flow diagram



(1) Activation of function
(2) Positive acknowledgement: PI service has been executed
(3) Reset function activation after receipt of acknowledgement
(4) Signal change by means of FB
(5) Not permissible
(6) Negative acknowledgement: Error has occurred, error code in the output parameter State

12.15.4.2 List of available PI services

The following PI services can be started from the PLC.

General PI services

PI service	Function
ASUB (Page 1029)	Assign interrupt
CANCEL (Page 1030)	Execute cancel
CONFIG (Page 1031)	Reconfiguration of tagged machine data
DIGION (Page 1031)	Digitizing on
DIGIOF (Page 1031)	Digitizing off
FINDBL (Page 1032)	Activate block search
LOGIN (Page 1032)	Activate password
LOGOUT (Page 1033)	Reset password
NCRES (Page 1033)	Trigger NC-RESET
SELECT (Page 1033)	Select program for processing for one channel
SETUDT (Page 1034)	Sets the current user data to active
SETUFR (Page 1034)	Activate user frame
RETRAC (Page 1035)	Retraction of the tool in the tool direction

PI services of tool management

PI service	Function
CRCEDN (Page 1035)	Create new cutting edge
CREACE (Page 1036)	Create tool cutting edge
CREATO (Page 1037)	Generate tool
DELECE (Page 1037)	Delete a tool cutting edge
DELETO (Page 1038)	Delete tool
MMCSEM (Page 1038)	Semaphores for various PI services
TMCRTO (Page 1039)	Create tool
TMFDPL (Page 1040)	Empty location search for loading
TMFPBP (Page 1042)	Empty location search
TMGETT (Page 1043)	T-number for the specified tool identifier with duplo number
TMMVTL (Page 1044)	Prepare magazine location for loading, unload tool
TMPOSM (Page 1046)	Position magazine location or tool
TMPCIT (Page 1047)	Set increment value for workpiece counter
TMRASS (Page 1047)	Reset active status
TRESMO (Page 1048)	Reset monitoring values
TSEARC (Page 1049)	Complex search using search screen forms
TMCRMT (Page 1052)	Create multitool
TMDLMT (Page 1053)	Delete multitool
POSMT (Page 1053)	Position multitool
FDPLMT (Page 1054)	Search/check an empty location within the multitool

12.15.4.3 PI service: ASUB

Function: Assign interrupt

A program stored on the NCK is assigned an interrupt signal of a channel. The program must be executable and the path and program name must be specified completely and correctly. For detailed information, please refer to:

References

Programming Manual, Job Planning; Section: "File and Program Management" > "Program Memory".

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.ASUB	Assign interrupt 1)
Unit	INT	1 ... 10	Channel
Addr1	STRING		Path name
Addr2	STRING		Program name
WVar1	WORD	1 ... 8	Interrupt number
WVar2	WORD	1 ... 8	Priority
WVar3	WORD	0/1	LIFTFAST 2)
WVar4	WORD	0/1	BLSYNC 3)
1) As an alternative for the interrupt assignment, the SETINT command can be used. See 2) 2) **References:** Programming Manual, Job Planning; Section: "Flexible NC programming" > "Interrupt routine (ASUB)" > "Fast retraction from the contour (SETINT, **LIFTFAST**, ALF)" 3) **References:** Programming Manual, Job Planning; Section: "Flexible NC programming" > "Interrupt routine (ASUB)" > "Assign and start interrupt routine (SETINT, PRIO, **BLSYNC**)"			

Note

The ASUB PI service must only executed in the RESET state of the specified channel.
References:
Programming Manual, Job Planning; Section: "Flexible NC-Programming" > "Interrupt routine (ASUB)"

12.15.4.4 PI service: CANCEL

Function: Execute Cancel

Triggers the "Cancel" function equivalent to the corresponding "Cancel alarm" button on the user interface (operator panel front).

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.CANCEL	Cancel

12.15.4.5 PI service: CONFIG

Function: Reconfiguration

The reconfiguration command activates machine data which has been entered sequentially by the operator or the PLC, almost in parallel.

The command can only be activated when the controller is in RESET state or the program is interrupted (NC stop at block limit). An FB 4 error checkback message is output if these conditions are not fulfilled (state = 3).

Parameterization

Parameterization			
Signal	**Type**	**Range of values**	**Meaning**
PIService	ANY	PI.CONFIG	Reconfiguration
Unit	INT	1	
WVar1	INT	1	Classification

12.15.4.6 PI service: DIGION

Function: Digitizing on

Selecting digitizing in the parameterized channel.

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.DIGION	Digitizing on
Unit	INT	1 to 10	Channel

12.15.4.7 PI service: DIGIOF

Function: Digitizing off

Deactivating digitizing in the parameterized channel.

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.DIGIOF	Digitizing off
Unit	INT	1 to 10	Channel

12.15.4.8 PI service: FINDBL

Function: Activate search

A channel is switched to search mode and the appropriate acknowledgement then transmitted. The search is then executed immediately by the NC. The search pointer must already be in the NC at this point in time. The search can be interrupted at any time by an NC RESET. Once the search is successfully completed, the normal processing mode is reactivated automatically. NC start then takes effect from the located search target.

It is the sole responsibility of the operator to ensure a collision-free approach path.

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI.FINDBL	block search	
Unit	INT	1 to 10	Channel	
WVar1	WORD	1, 2, 3	Preprocessing mode:	
			1	Without calculation
			2	With calculation
			3	With main block consideration

12.15.4.9 PI service: LOGIN

Function: Create password

Transfers the parameterized password to the NC. The passwords generally consist of eight characters. If required, blanks must be added to the string of the password.

Example

Password: STRING[8] := 'SUNRISE';

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.LOGIN	Create password
Unit	INT	1	NC
Addr1	STRING	8 characters	Password

12.15.4.10 PI service: LOGOUT

Function: Reset password

The password last transferred to the NC is reset.

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.LOGOUT	Reset password
Unit	INT	1	NC

12.15.4.11 PI service: NCRES

Function: Trigger NC-RESET

Triggers an NC-RESET. Parameters "Unit" and "WVar1" must always be set to 0.

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.NCRES	Trigger NC-RESET
Unit	INT	0	-
WVar1	WORD	0	-

12.15.4.12 PI service: SELECT

Function: Select processing for a channel

A program stored on the NC is selected for one channel for execution. This is possible only if the file may be executed. The path names and the program names are to be written in correct notation. For detailed information, please refer to:

References

Programming Manual, Job Planning; Section: "File and Program Management" > "Program Memory".

Possible block types

Block types	
Workpiece directory	WPD
Main program	MPF
Subprogram	SPF
Cycles	CYC
Asynchronous subprograms	ASP
Binary files	BIN

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.SELECT	Program selection
Unit	INT	1 ... 10	Channel
Addr1	STRING		Path name
Addr2	STRING		Program name

12.15.4.13 PI service: SETUDT

Function: Set function current user data active

The current user data such as tool offsets, basic frames and settable frames is set to active in the next NC block (only in Stop state).

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI.SETUDT	Activate user data	
Unit	INT	1 to 10	Channel	
WVar1	WORD	1 to 5	User data type	
			1	Active tool offset
			2	Active basic frame
			3	Active settable frame
			4	Active global basic frame
			5	Active global settable frame
WVar2	WORD	0	Reserved	
WVar3	WORD	0	Reserved	

12.15.4.14 PI service: SETUFR

Function: Activate user frames

User frames are loaded to the NC. All necessary frame values must be transferred to the NC first with FB 3 "Write variables".

Parameterization

Parameterization			
Signal	Type	Range of values	Meaning
PIService	ANY	PI.SETUFR	Activate user frames
Unit	INT	1 to 10	Channel

12.15.4.15 PI service: RETRAC

Function: Select JOG retract

Selects the JOG retract mode. The retraction axis or the geometry axis of the WCS with which the retraction is executed can be determined by the NC or specified explicitly.

Note

The PI service can only be activated in the "Reset" state in the JOG mode.

Parameter assignment

Parameter assignment				
Signal	**Type**	**Range of values**	**Meaning**	
PIService	ANY	PI.RETRAC	Select JOG retract mode	
Unit	INT	1 to 10	Channel	
WVar1	INT	0 to 3	Retraction axis:	
			0	Determination of the retraction axis by the NC.
				Case 1: JOG retract was already selected once, but not completed yet ⇒ The retraction axis selected last is selected again.
				Case 2: JOG retract is selected first time ⇒ The retraction axis is determined by the NC: - Standard: 3rd geometry axis - Grinding and turning tools: 1st geometry axis
				Note: The active retraction axis can be read via the OPI variable retractState.bit 2/3
			1	Retraction axis is 1st geometry axis of the WCS
			2	Retraction axis is 2nd geometry axis of the WCS
			3	Retraction axis is 3rd geometry axis of the WCS
WVar2	INT	0	Reserved. The value must be pre-assigned with 0.	

12.15.4.16 PI service: CRCEDN

Function: Creates new cutting edge

If the T number of an existing tool is entered in parameter "T Number" the PI service, then a tool edge for the existing tool is created (in this case, the parameter "D number", i.e. the number of the edge to be created, has a value range of 00001 - 00009).

If a positive T number is specified as a parameter and the tool for the T number entered does not exist, the PI service is aborted.

If a value of 00000 is entered as the T number (model of absolute D numbers), the D number values can range from 00001 - 31999. The new cutting edge is set up with the specified D number.

If the specified cutting edge already exists, the PI service is aborted in both cases.

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.CRCEDN	Create new cutting edge
Unit	INT	1 ... 10	TOA
WVar1	INT		T number of tool for which cutting edge must be created. A setting of 00000 states that the cutting edge should not refer to any particular tool (absolute D number).
WVar2	INT	1 ... 9 or 01 - 31999	Edge number of tool cutting edge

12.15.4.17 PI service: CREACE

Function: Create tool cutting edge

Creation of the cutting edge with the next higher / next unassigned D number for the tool with the transferred T number in TO, TS (if present). The cutting edge for the OEM cutting edge data is set up simultaneously in the TUE block (if one is present).

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.CREACE	Create tool cutting edge
Unit	INT	1 ... 10	TOA
WVar1	INT		T number

12.15.4.18 PI service: CREATO

Function: Create tool

Creation of a tool with specification of a T number. The tool is entered as existing in the tool directory area (TV). The first "cutting edge" D1 (with zero contents) is created for tool offsets in the TO block. D1 (with zero contents) is also created for the OEM "cutting edge" data in the TUE block - if one is present. If a TU block exists, it will contain the data set for the tool.

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.CREATO	Create tool
Unit	INT	1 ... 10	TOA
WVar1	INT		T number

12.15.4.19 PI service: DELECE

Function: Delete a tool cutting edge

If the T number of an existing tool is specified in parameter "T number" in the PI service, then a cutting edge is deleted for this particular tool (in this case, parameter "D number" (number of cutting edge to be created) has a value range of 00001 - 00009). If a positive T number is specified as a parameter and the tool for the T number entered does not exist, then the PI service is aborted. If a value of 00000 is entered as the T number (model of absolute D numbers), then the D number values can range from 00001 - 31999. If the specified cutting edge does not exist, then the PI service is aborted in both cases.

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.DELETE	Delete cutting edge
Unit	INT	1 ... 10	TOA
WVar1	INT		T number of tool for which cutting edge must be created. A setting of 00000 states that the cutting edge should not refer to any particular tool (absolute D number).
WVar2	INT	1 ... 9 or 01 ... 31999	Edge number of cutting edge that must be deleted

12.15.4.20 PI service: DELETO

Function: Delete tool

Deletes the tool assigned to the transferred T number with all cutting edges (in TO, in some cases TU, TUE and TG (type 4xx), TD and TS blocks).

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.DELETO	Delete tool
Unit	INT	1 ... 10	TOA
WVar1	INT		T number

12.15.4.21 PI service: MMCSEM

Semaphores for various PI services, for use by HMI and PLC

Ten semaphores are available for each channel. These protect critical functions for the HMI/PLC. If a function has a critical section with respect to the data to be read by the NC, several HMI/PLC units can synchronize by setting the semaphores with the corresponding function number.

Semaphores are managed by the HMI/PLC. A semaphore value of 1stipulates a Test & Set operation for the semaphores of the specified function number. The return value of the PI service represents the result of this operation:

- Checkback value Done := TRUE: Semaphore has been set, critical function can be called
- Checkback value Error := TRUE with state = 3: Semaphore was already set, critical function cannot be called at the present time. This must be repeated later.

NOTICE
On completion of the operation (reading data of this PI service) it is **essential** that the **semaphore** is **enabled again.**

Parameter

- WVar1 = <function number>

Function number	PI service
1	TMCRTO (create tool)
2	TMFDPL (search for empty location for loading)
3	TMMVTL (prepare magazine location for loading, unload tool)
4	TMFPBP (search for location)
5	TMGETT (search for tool number)
6	TSEARC (search for tool)
7 ... 10	Reserved

- WVar2 = <value>

Value	Meaning
0	Reset semaphore
1	Test and set semaphore

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.MMCSEM	Set semaphore
Unit	INT	1, 2 to 10	Channel
WVar1	INT	1 ... 10	FunctionNumber
WVar2	WORD	0, 1	SemaphoreValue

12.15.4.22 PI service: TMCRTO

Function: Create tool

Creating a tool with specification of the identifier, a duplo number, e.g. with:

- $TC_TP1[y] = duplo number;
- $TC_TP2[y] = "tool identifier";

Or optionally using a T number, e.g. with y = T number

The tool is entered as existing in the tool directory area (TV). The first cutting edge "D1" (with zero contents) is created for tool offsets in the TO block. "D1" (with zero contents) is also set up for the monitoring data in the TS block, and simultaneously with zero contents for the OEM cutting edge data in the TUE block - if one is present. The TD block contains the identifier, duplo number and number of cutting edges (=1) for the T number that is entered optionally or allocated by the NC.

If a TU block exists, it will contain the data block for the tool. After execution of the PI service, the T number of the tool created is available in the TV block under **TnumWZV**.

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Section "PI service: MMCSEM (Page 1038)".

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.TMCRTO	Create tool
Unit	INT	1 - 10	TOA
WVar1	INT		T number
WVar2	INT		Duplo number
Addr1	STRING	Max. 32 characters	Tool identifier
T number > 0 means a T number must be specified T number = -1 means that the NCK should allocate a T number The example shows T number = -1 ⇒ T number assigned by NCK			

12.15.4.23 PI service: TMFDPL

Function: Search for empty location for loading, depending on the parameter assignment

- Location_number_to = -1, Magazine_number_to = -1

  Searches all magazines in the specified area (= channel) for an empty location for the tool specified with a T number. After execution of the PI service, the magazine and locations numbers found during the search are listed in the configuration block of the channel (component **magCMCmdPar1** (magazine number) and **magCMCmdPar2** (location number)). Location_number_ID and magazine_number_ID can be set as search criteria or not (= -1). The PI service is acknowledged positively or negatively depending on the search result.

- Location_number_to = -1, Magazine_number_to = Magazine_number

  An empty location for the tool specified with a T number is searched for in the specified magazine. Location_number_ID and magazine_number_ID can be set as search criteria or not (= -1). The PI is acknowledged positively or negatively depending on the search result.

- Location_number_to = Location_number, Magazine_number_to = Magazine_number

  The specified location is checked, to confirm that it is free to be loaded with the specified tool. Location_number_ID and magazine_number_ID can be set as search criteria or not (= -1). The PI service is acknowledged positively or negatively depending on the search result.

Command parameters 1 and 2 are located at source.

Loading: If source is an internal loading magazine, then the command parameters are located at the target (a real magazine).

Unloading: Source is always a real magazine.

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Section "PI service: MMCSEM (Page 1038)".

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.TMFDPL	Empty location for loading
Unit	INT	1 - 10	TOA
WVar1	INT		T number
WVar2	INT		Location_number _to
WVar3	INT		Magazine_number_to
WVar4	INT		Location_number _ID
WVar5	INT		Magazine_number_ID

12.15.4.24 PI service: TMFPBP

Function: Empty location search

This service searches the specified magazine(s) for an empty location which meets the specified criteria such as tool size and location type.

If the search is successful, the result can be read from the following OPI variables:

- magCMCmdPar1 (magazine number)
- magCMCmdPar2 (location number)

NOTICE
The PI service can only be requested with FB 7. See Section "FB 7: PI_SERV2 (PI service request) (Page 1065)".

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Section "PI service: MMCSEM (Page 1038)".

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI.TMFPBP	Empty location search	
Unit	INT	1 to 10	TOA	
WVar1	INT		**Magazine number** of the magazine **from** which the search is to be performed	
WVar2	INT		**Location number** of the location in the magazine from WVar1	
WVar3	INT		**Magazine number** of the magazine up to which the search is to be performed	
WVar4	INT		**Location number** of the location in the magazine from WVar3	
WVar5	INT		Magazine number reference	
WVar6	INT		Location number reference	
WVar7	INT	0 to 7	Number of required half locations to left	
WVar8	INT	0 to 7	Number of required half locations to right	
WVar9	INT	0 to 7	Number of required half locations in upward direction	
WVar10	INT	0 to 7	Number of required half locations in downward direction	
WVar11	INT		Number of required location type	
WVar12	INT	0 to 4	Specifies the required search direction	
			0	Search strategy as set in $TC_MAMP2
			1	Forward
			2	Backward
			3	Symmetrical

Examples for setting of the search area

WVar1 : (from)	WVar2 : (location)	WVar3 : (to)	WVar4 : (location)	Search area
#M1	#P1	#M1	#P1	Only location #P1 in magazine #M1 is checked
#M1	#P1	#M2	#P2	Locations starting at magazine #M1, location #P1 up to magazine #M2, location #P2 are searched
#M1	-1	#M1	-1	All locations in magazine #M1 - and no others - are searched
#M1	-1	-1	-1	All locations starting at magazine #M1 are searched
#M1	#P1	-1	-1	All locations starting at magazine #M1 and location #P1 are searched
#M1	#P1	#M1	-1	Locations in magazine #M1 starting at magazine #M1 and location #P1 in this magazine are searched
#M1	#P1	#M2	-1	Locations starting at magazine #M1, location #P1 up to magazine #M2 are searched
#M1	-1	#M2	#P2	Locations starting at magazine #M1 up to magazine #M2, location #P2 are searched
#M1	-1	#M2	-1	Locations starting at magazine #M1 up to and including magazine #M2 are searched
-1	-1	-1	-1	All magazine locations are searched

12.15.4.25 PI service: TMGETT

Function: Determine T-number for the specified tool identifier with duplo number

Determining the T number for a specified tool identifier with duplo number. Whether a T number from the PI service was found, is displayed in the variable resultNrOfTools in the TF block. If the specified tool does not exist, then the number 0 is returned. If the number 1 is returned, then the T number is displayed in the variable resultToolNr in the TF block. Since the PI-service returns a result in the variable resultToolNr, the service is to be backed up using the semaphore mechanism (PI service _N_MMCSEM) with the function number for _N_TMGETT.

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Section "PI service: MMCSEM (Page 1038)".

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.TMGETT	Determining the T number
Unit	INT	1 ... 10	TOA
Addr1	STRING	Max. 32 characters	Name of the tool, for which the T number is searched
WVar1	INT		Duplo number of the tool, for which the T number is searched

12.15.4.26 PI service: TMMVTL

Function: Prepare magazine location for loading, unload tool

This PI service is used both to load and unload tools. Whether the PI service initiates a loading or unloading operation depends on the assignment between the real locations and the "from" parameters and "to" parameters: Loading ⇒ 'From' = Loading point/station, unloading ⇒ 'To' = loading point/station

The TMMVTL PI service is used for all motions.

1. Loading and unloading (loading point ↔ magazine)
2. Loading and unloading (loading point ↔ buffer storage, e.g. spindle)
3. Relocation within a magazine
4. Relocation between different magazines
5. Relocation between magazine and buffer storage
6. Relocation within buffer storage

The following variables from the TM block are used to monitor case 1, 3, 4, 5:

- magCmd (area no. = TO unit, line = magazine number)
- magCmdState <- "acknowledgement"

The following variables from the TMC block are used to monitor case 2, 6):

- magCBCmd (area no. = TO unit)
- magCBCmdState <- "acknowledgement"

Loading

Prepares the specified real magazine for the specified channel for loading, i.e. the magazine traverses to the selected location for loading at the specified loading point/station (location_number_from, magazine_number_from) and inserts the tool.

When location_number_to = -1, an empty location for the tool specified by a T number is first sought in the specified magazine and the magazine then traversed. After execution of the PI service, the number of the location found is listed in the TM area in component **magCMCmdPar2** for the **real** magazine of the channel.

With location_number_to = -2 and a valid magazine number, loading takes place into the currently queued magazine position of the specified magazine. After execution of the PI service, the number of the location for tool loading is listed in the TM area in component **magCMCmdPar2** for the real magazine of the channel.

Unloading

The tool specified by the tool number is unloaded at the specified loading point/station (location_number_to, magazine_number_to), i.e. the magazine is traversed to the position for unloading and the tool is then removed. The magazine location for the tool is marked as being free in the TP block. The tool can be specified either using a T number or by means of the location and magazine numbers. An unused specification has the value -1.

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Section "PI service: MMCSEM (Page 1038)".

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.TMMVTL	Make magazine location ready for loading, unloading tool
Unit	INT	1 ... 10	TOA
WVar1	INT		T number
WVar2	INT		Location_number_from
WVar3	INT		Magazine_number_from
WVar4	INT		Location_number_to
WVar5	INT		Magazine_number_to

12.15.4.27 PI service: TMPOSM

Function: Position magazine location or tool, depending on the parameter assignment

A magazine location, which has either been specified directly or qualified via a tool located on it, is traversed to a specified position (e.g. in front of a load location) via the PI service.

The PI service makes a magazine location, which can be qualified in various ways, traverse in front of a specified load location. The load location must be specified in the PI parameters "location number_from" and "magazine number_from" (compulsory!).

The magazine location to be traversed can be qualified by the following:

- T number of the tool

  The location where the tool is positioned traverses;
  the "tool identifier", "duplo number", "location number_from" and "magazine number_from" parameters are irrelevant (i.e. values "" , "-0001", "-0001","-0001").

- Tool identifier and duplo number

  The location where the tool is positioned traverses;
  the "T number", "location number_from" and "magazine number_from" parameters are irrelevant (i.e. value "-0001" each).

- Direct specification of the location in the "location_number_from" and "magazine_number_from" parameters

  The tool-qualifying parameters T number, "tool identifier" and "duplo number" are irrelevant (i.e. values "-0001", "", "-0001").

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.TMPOSM	Position magazine location or tool
Unit	INT	1 ... 10	TOA
Addr1	STRING	max. 32 characters	Tool identifier
WVar1	INT		T number
WVar2	INT		Duplo number
WVar3	INT		Location_number_from
WVar4	INT		Magazine_number_from
WVar5	INT		Location number_ref
WVar6	INT		Magazine number_ref

12.15.4.28 PI service: TMPCIT

Function: Set increment value for workpiece counter

Incrementing the workpiece counter of the spindle tool

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI.TMPCIT	Set increment value for workpiece counter
Unit	INT	1 ... 10	TOA
WVar1	WORD	0 ... max.	Spindle number; corresponds to the type index in the location data with spindle location type of the buffer magazine in channel.000 = main spindle
WVar2	WORD	0 ... max.	Increment value; indicates the number of spindle revolutions after which the workpiece counter is incremented

12.15.4.29 PI service: TMRASS

Function: Reset active status

Resetting the active status on worn tools
This PI service is used to search for all tools with the tool status active and disabled. The active status is then canceled for these tools. Potentially appropriate times for this PI service are the negative edge of NC/PLC interface signal "tool disable ineffective", an end of program, or a channel RESET. This PI service is intended mainly for the PLC, since it knows when the disabled tool is finally no longer to be used.

Parameterization

Signal	Type	Range of values	Meaning
PIService	ANY	PI. TMRASS	Reset active status
Unit	INT	1 ... 10	TO area

12.15.4.30 PI service: TRESMO

Function: Reset monitoring values

This PI service resets the monitoring values of the designated edges of the designated tools to their setpoint (initial) values.
This is only performed for tools with active monitoring.

See also the RESETMON command.

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI. TRESMO	Reset monitoring values	
Unit	INT	1 ... 10	TO area	
WVar1	WORD	- max ... max	ToolNumber	
			0:	Applies to all tools
			> 0:	Applies only to this tool
			< 0:	Applies to all sister tools of the specified T No.
WVar2	WORD	0 ... max.	D number	
			< 0:	Monitoring of specified edge of specified tools is reset.
			0:	Monitoring of all edges of specified tools is reset.
WVar3	WORD	0 ...15	Monitoring types Type of monitoring to be reset. This parameter is binary-coded.	
			1:	Tool-life monitoring is reset.
			2:	Count monitoring is reset.
			4:	Wear monitoring is reset.
			8:	Sum-offset monitoring is reset. Combinations of monitoring types can be reset by adding the values above.
			0:	All active tool-monitoring functions ($TC_TP9) are reset.

12.15.4.31 PI service: TSEARC

Function: Complex search using search screen form, depending on the parameter assignment

The PI service allows you to search for tools with specified properties within a search domain (in one or more magazines starting and ending at a specific location). The specified properties refer only to data of the tools and their cutting edges.

The PI service is only available if tool management is activated.

You can define a search direction and the number of hits for the PI service (e.g. one tool for the next tool with matching properties or all tools with the specified properties).

As a result of this service, the user who made the call receives a list of the internal T numbers of the tools found.

The search criteria can only be specified as AND operation. If an application needs to define an OR operation for the search criteria, it must first execute a series of queries with AND criteria and then combine/evaluate the results of the individual queries.

To assign the parameters of the PI service, the properties of the required tools are first defined via variable service in the TF block. For this purpose, the relevant comparison criteria (which tool data is to be compared?) are highlighted in the TF block in the operand screen forms (parMaskT..), the comparison operator data (parDataT..) assigned the appropriate comparison types to be executed (==, <, >, < =, > =, &&) and the comparison values entered in the operand data.

The PI service is then initiated and, after its successful return, the variable service from the TF block is used to read out the number of hits in the variable resultNrOfTools and the result list in the variable resultToolNr (i.e. the list of internal T numbers of the tools found in the search (resultNrOfTools quantity)).

The PI service must be encapsulated with a semaphore from its preparation until the successful return of the result. This is the only way to ensure exclusive access and the exclusive use of the TF block in conjunction with the TSEARC PI service (see MMCSEM PI service).

If the service is configured incorrectly, a malfunction occurs. In all other cases, it will return a result, even if no tools are found (resultNrOfTools = 0).

The search domain can be defined as follows in the parameters "MagNrFrom", "PlaceNrFrom", "MagNrTo", "PlaceNrTo":

MagNr From	PlaceNr From	MagNr To	PlaceNr To	Search area
WVar1	WVar2	WVar3	WVar4	
#M1	#P1	#M2	#P2	Locations starting at magazine #M1, location #P1 up to magazine #M2, location #P2 are searched
#M1	-1	#M1	-1	All locations in magazine #M1 - and no others - are searched
#M1	-1	-1	-1	All locations starting at magazine #M1 are searched
#M1	#P1	-1	-1	All locations starting at magazine #M1 and location #P1 are searched
#M1	#P1	#M1	-1	Locations in magazine #M1 starting at magazine #M1 and location #P1 in this magazine are searched
#M1	#P1	#M2	-1	Locations starting at magazine #M1, location #P1 up to magazine #M2 are searched
#M1	-1	#M2	#P2	Locations starting at magazine #M1 up to magazine #M2, location #P2 are searched
#M1	-1	#M2	-1	Locations starting at magazine #M1 up to and including magazine #M2 are searched
-1	-1	-1	-1	All magazine locations are searched

For a symmetric search (see "SearchDirection" parameter), the search area may stretch over only a single magazine (cases 2 and 5 from the above table). If another search domain is specified, the service will malfunction. A reference location must be entered in the parameters "MagNrRef" and "PlaceNrRef", with respect to which the symmetric search is done.

The reference location is a buffer location (a location from the magazine buffer, i.e. change position, gripper, etc.) or a load point (a location from the internal loading magazine). The search is executed symmetrically with reference to the magazine location in front of the specified reference location. A multiple assignment to the magazine being searched must be configured in the TPM block for the reference location. If this is not the case, a malfunction occurs. If the magazine location in front of the reference location is outside the search domain, the service responds as if it has not found a matching location.

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Section "PI service: MMCSEM (Page 1038)".

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI.TSEARC	Complex search using search screen forms	
Unit	INT	1 ... 10	TOA	
WVar1	INT		MagNrFrom Magazine number of magazine from which search must begin	
WVar2	INT		PlaceNrFrom Location number of location in magazine MagNrFrom, at which search must begin	
WVar3	INT		MagNrTo Magazine number of magazine at which search must end	
WVar4	INT		PlaceNrTo Location number of location in magazine MagNrTo, at which search must end	
WVar5	INT		MagNrRef Magazine number of (internal) magazine, with reference to which the symmetrical search is to be performed. (this parameter is only relevant with a "symmetrical" search direction)	
WVar6	INT		PlaceNrRef Location number of location in magazine MagNrRef, with reference to which the symmetrical search is to be performed. This parameter is only relevant with a "symmetrical" search direction	
WVar7	INT	1, 2, 3	SearchDirection specifies the required search direction.	
			1:	Forwards from the first location of the search domain
			2:	Backwards from the last location of the search domain
			3:	symmetric to the real magazine location, which is before the location specified with Magazine-Number_ID and Location-Number_ID
WVar8	INT	0, 1, 2, 3	KindofSearch	
			0:	Find all tool with this property cutting edge specifically
			1:	Search for the first tool found with this property (cutting edge specifically)
			2:	Browse all cutting edges to find all tool with this property
			3:	Browse all tools to search for the first tool found with this property

12.15.4.32 PI service: TMCRMT

Function: Create multitool

Creating a new multitool with specification of an identifier, optionally a multitool number, number of locations and type of distance coding. Tool T numbers, magazine numbers and multitool numbers are unique in the permissible number range 1 ... 32000. When you create the multitool, all of the associated locations are also generated.

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Section "PI service: MMCSEM (Page 1038)".

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI.TMCRMT	Create multitool	
Unit	INT	1 ... 10	TOA	
Addr1	STRING	Max. 32 characters	Multitool identifier	
WVar1	INT	0	Reserved	
WVar2	INT	-1 ... 32000	Multitool number	
			1 to 32000:	Multitool number specified by the user
			-1:	NCK allocates the multitool number itself
WVar3	INT	2 ... MD17504 $MN_MAX_ TOOLS_PER_ MULTITOOL	Number of locations in the multitool	
WVar4	INT	1 ... 3	Type of distance coding	

12.15.4.33 PI service: TMDLMT

Function: Delete multitool

Deletes the multitool in all data blocks in which it is stored. Tools equipped with multitool are then no longer equipped and no longer loaded, but they are still defined.

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI.TMDLMT	Delete multitool	
Unit	INT	1 ... 10	TOA	
Addr1	STRING	Max. 32 characters	Multitool identifier	
WVar1	INT	0	Reserved	
WVar2	INT	-1 ... 32000	Multitool number	
			1 to 32000:	Delete the specified multitool number (Addr1 is not evaluated)
			-1:	Delete the multitool with the name specified in Addr1
WVar3	INT	0, 1	Delete any tools contained	
			0:	Do not delete
			1:	Delete

12.15.4.34 PI service: POSMT

Function: Position multitool

Positions the multitool at the programmed location or alternatively at the programmed tool, which is located in one of the locations of the multitool. The tool itself can either be specified using its T number or with its name plus duplo number. A multitool can only be positioned if it is at a toolholder location and if no tool offset with regard to this toolholder is active.

Positioning is programmed by programming WVar1 (= the toolholder whose multitool should be positioned) and either:

- With WVar2 (tool number), or
- With Addr1 and WVar3 (tool identifier/duplo number), or
- With WVar4 (multitool location number).

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI.POSMT	Position multitool	
Unit	INT	1 ... 10	TOA	
Addr1	STRING	Max. 32 characters	Tool identifier of tool to be positioned in the multitool **Note:** If no tool identifier is specified, then an empty string must be entered (then WVar2 must be programmed)	
WVar1	INT	1 ... 999	Number of the toolholder	
WVar2	INT	-1 ... 32000	Tool number (T number) of the tool to be positioned in the multitool	
			-1:	The tool number is irrelevant (Addr1 and WVar3 must then be programmed)
WVar3	INT	-1 ... 32000	Duplo number of the tool to be positioned in the multitool	
			-1:	Duplo number is irrelevant (then WVar2 must be programmed)
WVar4	INT	1 ... 999	Multitool location number of the location to which the system should be positioned	

12.15.4.35 PI service: FDPLMT

Function: Search/check an empty location within the multitool, depending on the parameter assignment

Searches in the multitool for a free location to accept the specified tool or checks the specified location in the multitool whether it is free for accepting the specified tool. The tool can either be specified using its T number, or alternatively, using its identifier and its duplo number.

Calling up the PI service is acknowledged positively or negatively depending on the search result.

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Section "PI service: MMCSEM (Page 1038)".

Parameterization

Signal	Type	Range of values	Meaning	
PIService	ANY	PI.FDPLMT	Search/check an empty location within the multitool	
Unit	INT	1 ... 10	TOA	
Addr1	STRING	Max. 32 characters	Tool identifier of tool to be positioned in the multitool **Note:** If no tool identifier is specified, then an empty string must be entered (then WVar2 must be programmed)	
WVar1	INT	-1 ... 32000	Tool number (T number) of the tool to be positioned in the multitool	
			-1:	The tool number is irrelevant (then Addr1 and WVar2 must be alternatively programmed)
WVar2	INT	-1 ... 32000	Duplo number of the tool to be positioned in the multitool	
			-1:	Duplo number is irrelevant (then WVar1 must be alternatively programmed)
WVar3	INT	-1 ... 32000	Number of the multitool in which the empty location search/testing should take place	
			-1:	Search across all multitools for an empty location – or check across all multitools whether the programmed location is free in one of them to accept the programmed tool
WVar4	INT	-1 ... 999	Multitool location number of the location to which the system should be positioned	
			-1:	Do not search for an empty location, but search for an empty location within the multitool

Call example

Program selection in channel 1 (main program and workpiece program)

Entry of PI service for DB 16 and STR for DB 124 with the S7 SYMBOL editor:

Parameterization		
Symbol	**Operand**	**Data type**
PI	DB 16	DB 16
STR	DB 124	DB 124

```
DATA_BLOCK DB 126           //Unassigned user DB, as instance for FB 4
FB 4
BEGIN
END_DATA_BLOCK

DATA_BLOCK DB 124
   struct
       PName:          string[32]:= '_N_TEST_MPF ';
       Path:           string[32]:= '/_N_MPF_DIR/';   //Main program
       PName_WST:      string[32]:= '_N_ABC_MPF';
       Path_WST:       string[32]:=                   //Workpiece program
                       '/_N_WCS_DIR/_N_ZYL_WPD';
   end_struct
BEGIN
END_DATA_BLOCK
Function FC "PICall" : VOID
   U   I 7.7;          //Unassigned machine control panel key
   S   M 0.0;          //Activate req.
   U   M 1.1;          //Done completed message
   R   M 0.0;          //Terminate job
   U   I 7.6;          //Manual error acknowledgement
   U   M 1.0;          //Error pending
   R   M 0.0;          //Terminate job

   CALL FB 4, DB 126 (
       Req :=          M0.0,
       PIService :=    PI.SELECT,
       Unit :=         1,                             // CHAN 1
       Addr1 :=        STR.Path,
       Addr2 :=        STR.PName,                     //Main-program selection
                       //Addr1:=STR.Path_WST,
                       //Addr2:=STR.PName_WST,        //Workpiece-program selection
       Error :=        M1.0,
       Done :=         M1.1,
       State :=        MW2);
```

12.15.5 FB 5: GETGUD read GUD variable

Function

The PLC user program can read a GUD variable (GUD = Global User Data) from the NCK or channel area using the FB GETGUD.

The FB has multi-instance capability. The call is permitted only in cyclic program OB1. An assignment for all parameters with Req = 0 is also permitted in OB 100. Capital letters must be used for the names of GUD variables.

Every FB 5 call must be assigned a separate instance DB from the user area.

A job is started when **FB 5 is called** by means of a positive edge change at control input Req. This job includes the name of the GUD variable to be read in parameter "Addr" with data type "STRING". The pointer to the name of the GUD variables is assigned symbolically to the "Addr" parameter with <DataBlockName>.<VariableName>. Additional information about this variable is specified in parameters "Area", "Unit", "Index1" and "Index2" (see table of block parameters).

When parameter "CnvtToken" is activated, a variable pointer (token) can be generated for this GUD variable as an option. This pointer is generated via the NC-VAR selector for system variables of the NC. Only this method of generating pointers is available for GUD variables. Once a pointer has been generated for the GUD variable, then it is possible to read and write via FB 2 and FB 3 (GET, PUT) with reference to this variable pointer. This is the only method by which GUD variables can be read. When FB 2 or FB 3 is parameterized, only parameter Addr1 ... Addr8 need to be parameterized for this GUD variable pointer. GUD variable fields are an exception. In these fields Line1 to Line8 must also be parameterized with the field device from these variables. The successful completion of the read process is displayed by the status parameter "Done"=TRUE.

The read process extends over several PLC cycles (generally 1 to 2).

Any errors are displayed via the output parameters "Error" and "State".

Note

In order to read a double variable from the NCK without adapting the format, an ANY pointer of the REAL 2 type must be specified in the target area for read data (e.g.: P#M100.0 REAL 2). If the basic program recognizes REAL 2 as the target type when reading a "double" variable, the data is applied to the PLC data area as a 64-bit floating point number.

FB 5 can only write GUD variables if basic program parameter "NCKomm" has been set to "TRUE" (in OB 100: FB 1, DB 7; see "FB 1: RUN_UP Basic program, startup section (Page 999)").

Declaration of the function

```
FUNCTION_BLOCK FB 5            //Server name
    KNOW_HOW_PROTECT
    VERSION : 3.0
VAR_INPUT
    Req :           BOOL;
    Addr:           ANY ;      //Variables name string
    Area            BYTE ;     //Area: NCK = 0, channel = 2
    Unit :          BYTE ;
    Index1:         INT;       //Field index 1
    Index2:         INT;       //Field index 2
    CnvtToken:      BOOL;      //Conversion into 10-byte token
    VarToken        ANY ;      //Struct with 10 bytes for the variable token
END_VAR

VAR_OUTPUT
    Error :         BOOL;
    Done :          BOOL;
    State :         WORD ;
END_VAR

VAR_IN_OUT
    RD:             ANY ;
END_VAR

BEGIN
END_FUNCTION_BLOCK
```

Description of formal parameters

The table below lists all formal parameters of the GETGUD function.

Signal	Type	Type	Value range	Comment	
Req	I	BOOL		Job start with positive signal edge	
Addr	I	ANY	[DBName].[VarName]	GUD variable name in a variable of data type STRING	
Area	I	BYTE		Area address:	
				0:	NCK variables
				2:	Channel variables
Unit	I	BYTE		NCK area: Unit:=1 Channel area: Channel no.	
Index1	I	INT		Field index 1 of variable Variable has the value 0 if no field index is used.	
Index2	I	INT		Field index 2 of variable Variable has the value 0 if no field index is used.	
CnvtToken	I	BOOL		Activate generation of a 10 byte variable token	
VarToken	I	ANY	[DBName].[VarName]	Address to a 10byte token (see example)	
Error	O	BOOL		Negative acknowledgment of job or execution of job impossible	
Done	O	BOOL		Job successfully executed	
State	O	WORD		See error identifiers	
RD	I/O	ANY	P#Mm.n BYTE x... P#DBnr.dbxm.n BYTE x	data to be read	

Error identifiers

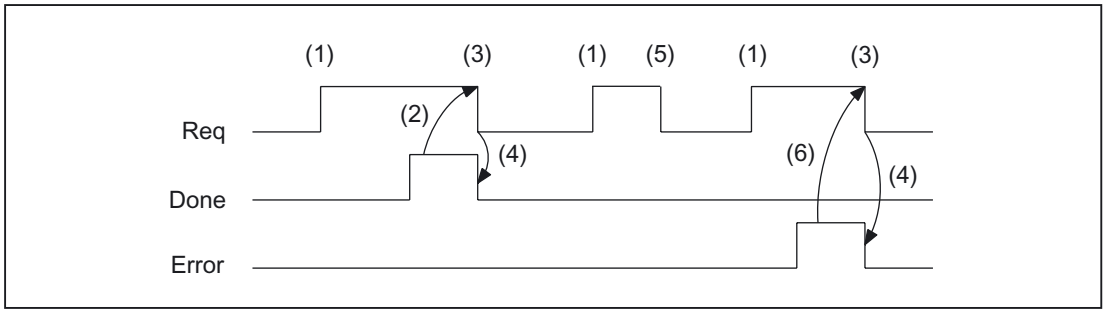
If it was not possible to execute a job, the failure is indicated by "logic 1" on status parameter error. The error cause is coded at the block output State:

State		Meaning	Note
WORD H	WORD L		
0	1	Access error	
0	2	Error in job	Incorrect compilation of Var. in a job
0	3	Negative acknowledgment, job not executable	Internal error, try: NC RESET
0	4	Data areas or data types do not tally	Check data to be read in RD
1	4	Insufficient local user memory available	read variable is longer than specified in RD
0	6	FIFO full	Job must be repeated, since queue is full
0	7	Option not set	BP parameter "NCKomm" is not set
0	8	Incorrect target area (SD)	RD may not be local data
0	9	Transmission occupied	Job must be repeated
0	10	Error in addressing	Unit contains value 0
0	11	Address of variable invalid	Address check (or variable name), area, unit
1 ... 8	13 (0x0d)	ANY data reference incorrect	String/NcVar data required has not been parameterized
0	15 (0x0f)	String more than 32 characters	GUD variable name too long

Configuration steps

To be able to read a GUD variable, its name must be stored in a string variable. The data block with this string variable must be defined in the symbol table so that the "Addr" parameter can be assigned symbolically for FB GETGUD. A structure variable can be defined optionally in any data area of the PLC to receive the variable pointer (see specification in following example).

Pulse diagram



(1) Activation of function
(2) Positive acknowledgment: variables have been written
(3) Reset function activation after receipt of acknowledgment
(4) Signal change by means of FB
(5) Not permissible
(6) Negative acknowledgment: Error has occurred, error code in the output parameter "State"

Call example 1

Read a GUD variable from channel 1 with the name "GUDVAR1" (type definition of the variables: INTEGER). The user-defined variable should be converted in a 10-byte variable pointer for the subsequent writing with the F3 (see also the table "Assignment of the data types" in "FB 2: Read GET NC variable (Page 1008) ").

Call and parameterization of FB 5 with instance DB 111:

```
// Data block for GUD variable
DATA_BLOCK DB_GUDVAR      //Assignment to symbol table

STRUCT
   GUDVar1 : STRING[32] := 'GUDVAR1';        //Name is defined by user
   GUDVar1Token :
   STRUCT
       SYNTAX_ID : BYTE ;
       area_and_unit : BYTE ;
       column : WORD ;
       line : WORD ;
       block type : BYTE ;
       NO. OF LINES : BYTE ;
       type : BYTE ;
       length : BYTE ;
   END_STRUCT;
END_STRUCT;

BEGIN
```

```
END_DATA_BLOCK

//Unassigned user DB, as instance for FB 5
DATA_BLOCK DB 111

         FB 5
BEGIN
END_DATA_BLOCK

//Unassigned user DB, as instance for FB 3
DATA_BLOCK DB 112

         FB 3
BEGIN
END_DATA_BLOCK

//A user-defined channel variable from channel 1 must be read
//with conversion into a variable pointer to allow subsequent
//writing of this variable.

Function FC "VariablenCall" : VOID
U      I 7.7;              //Unassigned machine control panel key
S      M 100.0;            //Activate req.
U      M 100.1;            //Done completed message
R      M 100.0;            //Terminate job
U      I 7.6;              //Manual error acknowledgment
U      M 102.0;            //Error pending
R      M 100.0;            //Terminate job

CALL FB 5, DB 111 (
             Req           := M 100.0,       //Starting edge for reading
             Area          := B#16#2,        //Channel variable
             Unit          := B#16#1,        //Channel 1
             Index1        := 0,             //No field index
             Index2        := 0,             //No field index
             CnvtToken     := TRUE,          //Conversion into 10-byte token
             VarToken      := DB_GUDVAR.GUDVar1Token,
             Error         := M 102.0,
             Done          := M 100.1,
             State         := MW 104
             RD            := P#DB99.DBX0.0 DINT 1      // free memory area
             );
```

After a successful FB-5 call the writing can be done via the returned address of the FB 5 parameter ("VarToken") with the help of FB3.

```
CALL FB 3, DB 112 (
             Req           := M 200.0,
             NumVar        := 1,             //Write 1 GUD variable
             Addr1         := DB_GUDVAR.GUDVar1Token,
             Error         := M 102.0,
             Done          := M 100.1,
             State         := MW 104
             SD1           := P#DB99.DBX0.0 DINT 1);
```

Call example 2

Read a GUD variable from channel 1 with the name "GUD_STRING" (type definition of the variables: STRING with length 30 bytes). The user-defined variable should be converted in a 10-byte variable pointer for subsequent writing with the FB 3.

Call and parameterization of FB 5 with instance DB 111:

```
// Data block for GUD variable
DATA_BLOCK DB_GUDVAR      //Assignment to symbol table

STRUCT
    GUDVarS : STRING[32] := 'GUD_STRING';        //Name is defined by user
    GUDVarSToken :
    STRUCT
       SYNTAX_ID : BYTE ;
       area_and_unit : BYTE ;
       column : WORD ;
       line : WORD ;
       block type : BYTE ;
       NO. OF LINES : BYTE ;
       type : BYTE ;
       length : BYTE ;
    END_STRUCT;

    string_of_GUD : STRING[30];         // must at least be so long as
                                        // the definition of 'GUD_STRING'!
    new_name : STRING[30] := 'GUD_123';
END_STRUCT;

BEGIN
END_DATA_BLOCK
```

```
//Unassigned user DB, as instance for FB 5
DATA_BLOCK DB 111

         FB 5
BEGIN
END_DATA_BLOCK

//Unassigned user DB, as instance for FB 3
DATA_BLOCK DB 112

         FB 3
BEGIN
END_DATA_BLOCK

//A user-defined channel variable from channel 1 must be read
//with conversion into a variable pointer to allow subsequent
//writing of this variable.

Function FC "VariablenCall" : VOID
U      I 7.7;              //Unassigned machine control panel key
S      M 100.0;            //Activate req.
U      M 100.1;            //Done completed message
R      M 100.0;            //Terminate job
U      I 7.6;              //Manual error acknowledgment
U      M 102.0;            //Error pending
R      M 100.0;            //Terminate job

CALL FB 5, DB 111 (
             Req           := M 100.0,      //Starting edge for reading
             Addr          := DB_GUDVAR.GUDVarS,
             Area          := B#16#2,       //Channel variable
             Unit          := B#16#1,       //Channel 1
             Index1        := 0,            //No field index
             Index2        := 0,            //No field index
             CnvtToken     := TRUE,         //Conversion into 10-byte token
             VarToken      := DB_GUDVAR.GUDVarSToken,
             Error         := M 102.0,
             Done          := M 100.1,
             State         := MW 104
             RD            := DB_GUDVAR.string_of_GUD);
```

After a successful FB-5 call the writing can be done via the returned address of the FB 5 parameter ("VarToken") with the help of FB3.

```
CALL FB 3, DB 112 (
             Req           := M 200.0,
             NumVar        := 1,             //Write 1 GUD variable
             Addr1         := DB_GUDVAR.GUDVarSToken,
             Error         := M 102.0,
             Done          := M 100.1
             State         := MW 104
             SD1           := DB_GUDVAR.new_name);
```

12.15.6 FB 7: PI_SERV2 (PI service request)

Function

Apart from a larger number of WVar parameters, the FB 7 function block has the same functionality as the FB 4. It is therefore recommended that you use the FB 7 function block instead of the FB 4.

For a detailed description of requesting a PI service, see Section "FB 4: PI_SERV PI service request (Page 1025)".

Declaration of the function

```
FUNCTION_BLOCK FB 7
Var_INPUT
    Req :            BOOL;
    PIService :      ANY ;
    Unit :           INT;
    Addr1 :          ANY ;
    Addr2 :          ANY ;
    Addr3 :          ANY ;
    Addr4 :          ANY ;
    WVar1 :          WORD ;
    WVar2 :          WORD ;
    WVar3 :          WORD ;
    WVar4 :          WORD ;
    WVar5 :          WORD ;
    WVar6 :          WORD ;
    WVar7 :          WORD ;
    WVar8 :          WORD ;
    WVar9 :          WORD ;
```

```
    WVar10 :        WORD ;
    WVar11 :        WORD ;
    WVar12 :        WORD ;
    WVar13 :        WORD ;
    WVar14 :        WORD ;
    WVar15 :        WORD ;
    WVar16 :        WORD ;
END_VAR
VAR_OUTPUT
    Error :         BOOL;
    Done :          BOOL;
    State :         WORD ;
END_VAR
```

Description of formal parameters

Signal	Type	Type	Range of values	Remark
Req	I	BOOL		Job request
PIService	I	ANY	[DBName].[VarName] Standard is: "PI".[VarName]	PI service description
Unit	I	INT	1...	Area number
Addr1 to Addr4	I	ANY	[DBName].[VarName]	Reference to strings specification according to selected PI service
WVar1 to WVar16	I	WORD	1...	Integer or word variables. Specification according to selected PI service
Error	O	BOOL		Negative acknowledgement of job or execution of job impossible
Done	O	BOOL		Job successfully executed
State	O	WORD		See error identifiers

12.15.7 FB 9: MtoN Control unit switchover

Function

This block allows switchover between several **control units** (HMI operator panel fronts and/or MCP machine control panels), which are connected to one or more NCU control modules via a bus system.

References:
Function Manual, Extended Functions; Several Control Panels on Multiple NCUs, Decentralized Systems (B3)

The **Interface** between the individual control units and the NCU (PLC) is the M : N interface in the data block DB 19. The FB 9 works with the signals of these interfaces.

Apart from initialization, sign-of-life monitoring and error routines, the following **basic functions** are also performed by the block for control unit switchover:

Tabulated overview of functions:	
Basic function	**Significance**
HMI queuing	HMI wants to go online with an NCU
HMI coming	HMI is connecting to an NCU
HMI going	HMI is disconnecting from an NCU
Forced break	HMI must break connection with 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 HMI

Brief description of a few important functions

Active/passive operating mode:

An online HMI can operate in two different modes:

Active mode:	Operator can control and monitor
Passive mode:	Operator can monitor (HMI header only)

After switchover to an NCU, this initially requests active operating mode in the PLC of the online NCU. If two control units 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 HMI at the press of a button.

MCP switchover

As an option, an MCP assigned to the HMI 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 on the HMI and "MCPEnable" must be set to TRUE. The MCP of the passive HMI is deactivated so that there is only ever one active MCP on an NCU at one time.

Boot condition

To prevent the previously selected MCP being reactivated 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.

Approvals

When one MCP is switched over to another, any active feed or axis enables will be retained.

Note

Keys actuated at the moment of switchover remain operative until the new MCP is activated (by the HMI, 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 example "Override Changeover").

The call is permitted only in cyclic program OB1.

Declaration of the function

```
FUNCTION_BLOCK FB 9
VAR_INPUT
    Ack :             BOOL ;            //Acknowledge interrupts
    OPMixedMode:      BOOL:= FALSE;     //Mixed operation with non-M-to-N-enabled
                                        OP //deactivated
    ActivEnable:      BOOL:= TRUE;      // Not supported
    MCPEnable :       BOOL:= TRUE;      // Activate MCP switchover
END_VAR
VAR_OUTPUT
    Alarm1 :          BOOL ;            // Interrupt: Error in HMI bus address, bus
                                        type!
    Alarm2 :          BOOL ;            // Interrupt: No confirmation HMI 1
                                        offline!
    Alarm3 :          BOOL ;            // Interrupt: HMI 1 is not going offline!
    Alarm4 :          BOOL ;            // Interrupt: No confirmation HMI 2
                                        offline!
    Alarm5 :          BOOL ;            // Interrupt: HMI 2 is not going offline!
    Alarm6 :          BOOL ;            // Interrupt: Queuing HMI is not going
                                        online!
    Report :          BOOL ;            // Message: Signoflife monitoring
    ErrorMMC :        BOOL ;            // Error detection HMI
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the M:N function.

Formal parameters of M:N function				
Signal	**Type**	**Type**	**Remark**	
Ack	I	BOOL	Acknowledge interrupts	
OPMixedMode	I	BOOL	Mixed operation deactivated for OP without M:N capability	
ActivEnable	I	BOOL	Function is not supported. Control panel switchover Interlocking via MMCx_SHIFT_LOCK in DB 19	
MCPEnable	I	BOOL	Activate MCP switchover:	
			TRUE:	MCP is switched over with operator panel.
			FALSE:	MCP is not switched over with operator panel. This can be used to permanently link an MCP. See also MMCx_MSTT_SHIFT_LOCK in DB 19.
Alarm1	Q	BOOL	Interrupt: Error in HMI bus address, bus type!	
Alarm2	Q	BOOL	Interrupt: No confirmation HMI 1 offline!	
Alarm3	Q	BOOL	Interrupt: HMI 1 is not going offline!	
Alarm4	Q	BOOL	Interrupt: No confirmation HMI 2 offline!	
Alarm5	Q	BOOL	Interrupt: HMI 2 is not going offline!	
Alarm6	Q	BOOL	Interrupt: Queuing HMI is not going online!	
Report	Q	BOOL	Message: Sign-of-life monitoring HMI	
ErrorMMC	Q	BOOL	Error detection HMI	

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 multi-instance-capable.

Example of a call for FB 9:

```
CALL FB 9, DB 109 (
Ack               := Error_ack,         //e.g., MCP RESET
OPMixedMode       := FALSE,
ActivEnable       := TRUE,
MCPEnable         := TRUE);             // Enable for MCP switchover
```

Note

Input parameter “MCPEnable” must be set to TRUE to enable the MCP switchover. The default value of these parameters is set in this way and need not be specially assigned when the function is called.

Interrupts, errors

The output parameters "Alarm1" to "Alarm6" and "Report" exist as information in the PLC and are output in the event of M:N errors visualized on the HMI by the appearance of alarms 410900 - 410906.

If execution of an HMI function has failed (and an appropriate error message cannot be displayed), status parameter "ErrorMMC" is set to 'logical 1' (e.g., booting error, when no connection is made).

Call example for FB 1 (Call in OB 100)

```
CALL "RUN_UP", "gp_par" (
    MCPNum                  := 1,
    MCP1In                  := P#I 0.0,
    MCP1Out                 := P#Q 0.0,
    MCP1StatSend            := P#Q 8.0,
    MCP1StatRec             := P#Q 12.0,
    MCP1BusAdr              := 255,               // Address of 1st MCP
    MCP1Timeout             := S5T#700MS,
    MCP1Cycl                := S5T#200MS,
    MCP1Stop                := TRUE,              // MCP switched off
    NCCyclTimeout           := S5T#200MS,
    NCRunupTimeout          := S5T#50S);
```

Example Override switchover

```
// Auxiliary flags used M100.0, M100.1, M100.2, M100.3
//Edge positive of MCP1Ready must check the override
//and measures for activation
// Initiate MCP block
//This example applies to the feedrate override.
//The interface and input bytes must be exchanged for spindle override.
U      DB10.DBX 104.0;       //MCP1Ready
EN     M    100.0;           //Edge trigger flag 1
JCN smth1;
S      M    100.2;           //Set auxiliary flag 1
R      M    100.3;           //Reset auxiliary flag 2
```

```
// Save override
      L DB21.DBB 4;          //Feed override interface
      T EB 28;               //Buffer storage (freely input
                             // or flag byte)

wei1:
U     M    100.2;           //Switchover takes place
O     DB10.DBX 104.0;       //MCP1Ready
JCN smth2;
U     DB10.DBX 104.0;       //MCP1Ready
FP    M    100.1;           //Edge trigger flag 2
JC smth2;
U     M    100.2;           //Switchover takes place
```

```
R     M    100.2;           //Reset auxiliary flag 1
JC smth2;
U     M    100.3;           //Comparison has taken place
SPB MCP;                    //Call MCP program
// Route the stored override to the interface of the switched MCP
// until the override values match
      L EB 28;              //Buffer storage open
      T DB21.DBB 4;         //Route override interface
      L EB 3;               //Override input byte for feed
      <>i;                  //Match?
JC smth2;                   //No, jump
S     M    100.3;           //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;
```

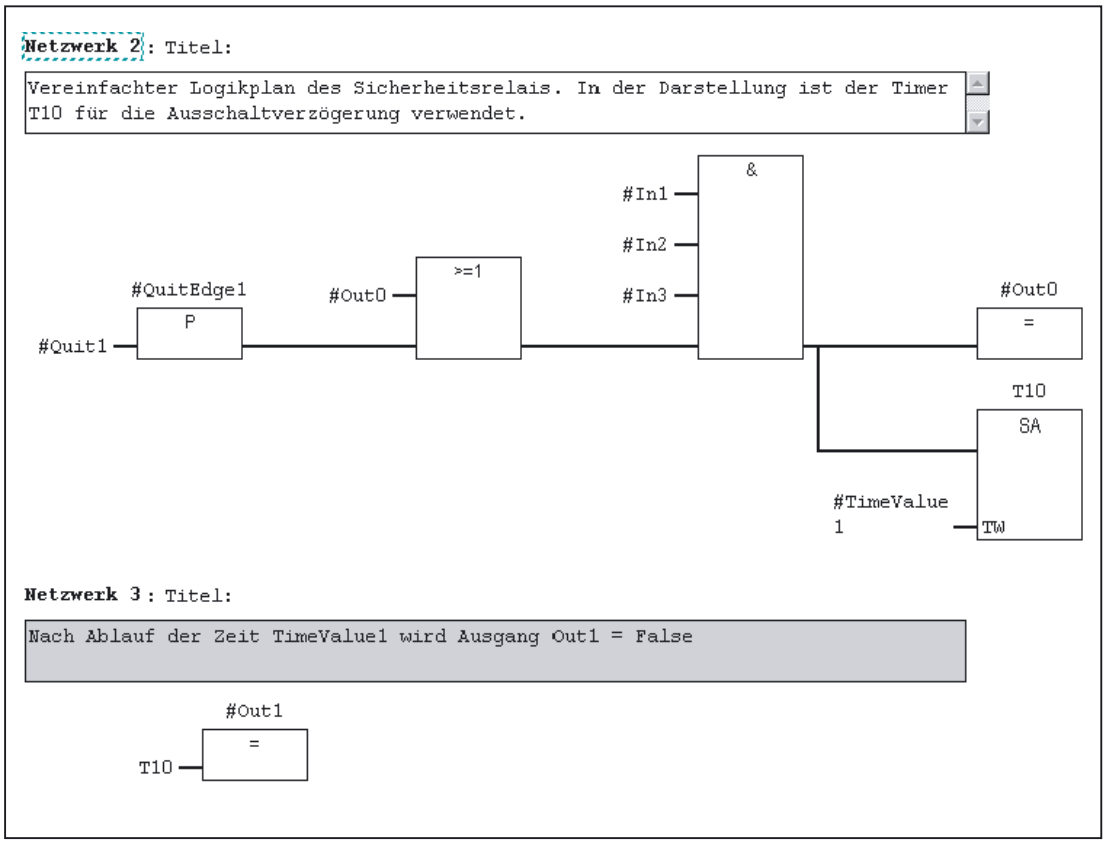
12.15.8 FB 10: Safety relay (SI relay)

Function

The SPL block "Safety relay" for "Safety Integrated" is the PLC equivalent of the NC function of the same name. The standard SPL "Safety relay" block is designed to support the implementation of an emergency stop function with safe programmable logic. However, it can also be used to implement other similar safety functions, e.g., control of a protective door. The function contains 3 input parameters (In1, In2, In3). On switchover of one of these parameters to the value 0, the output Out0 is deactivated without delay and outputs Out1, Out2 and Out3 deactivated via the parameterized timer values (parameters TimeValue1, TimeValue2, TimeValue3). The outputs are activated again without delay, if inputs In1 to In3 take the value 1 and a positive edge change is detected at one of the acknowledgement inputs Ack1, Ack2. To bring the outputs to their basic setting (values = 0) after booting, the parameter "FirstRun" must be configured as follows. The parameter "FirstRun" must be switched to the value TRUE via a retentive data (memory bit, bit in data block) on the 1st run after control booting. This data can be preset, e.g., in OB 100. The parameter is reset to FALSE when FB 10 is executed for the first time. Separate data must be used for parameter "FirstRun" for each call with its own instance.

Simplified block diagram in CSF

The figure below shows only one acknowledgment input Ack1 and one delayed deactivation output Out1. The circuit for Ack2 and the other delayed outputs are identical. The parameter FirstRun is also missing in the function diagram. The mode of operation is described above.



Declaration of the function

```
FUNCTION_BLOCK FB 10
VAR_INPUT
    In1 : BOOL       := TRUE;            //Input 1
    In2 : BOOL       := TRUE;            //Input 2
    In3 : BOOL       := TRUE;            //Input 3
    Ackn1 :          : BOOL;             //Ack 1 signal
    Ackn2 :          :BOOL;              //Ack 2 signal
    TimeValue1 :     TIME := T#0ms ;     //TimeValue for output 1
    TimeValue2 :     TIME := T#0ms ;     //TimeValue for output 2
    TimeValue3 :     TIME := T#0ms ;     //TimeValue for output 3
END_VAR
```

```
VAR_OUTPUT
    Out0            : BOOL;             //Output without delay
    Out1            : BOOL;             //Delayed output to false by timer 1
    Out2            : BOOL;             //Delayed output to false by timer 2
    Out3            : BOOL;             //Delayed output to false by timer 3
END_VAR
VAR_INOUT
    FirstRun        : BOOL;             //TRUE by user after 1st start of SPL
END_VAR
```

Description of formal parameters

The following table shows all formal parameters of the SI relay function:

Formal parameters of SI relay function			
Signal	**Type**	**Type**	**Remark**
In1	I	BOOL	Input 1
In2	I	BOOL	Input 2
In3	I	BOOL	Input 3
Ackn1	I	BOOL	Acknowledge input 1
Ackn2	I	BOOL	Acknowledge input 2
TimeValue1	I	TIME	Time value 1 for OFF delay
TimeValue2	I	TIME	Time value 2 for OFF delay
TimeValue3	I	TIME	Time value 3 for OFF delay
Out0	O	BOOL	Output, instantaneous (no delay)
Out1	O	BOOL	Output, delayed by TimeValue1
Out2	O	BOOL	Output, delayed by TimeValue2
Out3	O	BOOL	Output, delayed by TimeValue3
FirstRun	I/O	BOOL	Activation of basic setting

Note

The block must be called once by the user program (per SI relay) cyclically in the OB1 cycle from when the SPL program starts. The user must provide an instance DB with any number for this purpose. The call is multi-instance-capable.

12.15.9 FB 11: Brake test

Function

The braking operation check should be used for all axes, which must be prevented from moving in an uncontrolled manner by a holding brake. This check function is primarily intended for the socalled "vertical axes".

The machine manufacturer can use his PLC user program to close the brake at a suitable moment in time (guide value every 8 hours, similar to the SI test stop) and allow the drive to produce an additional torque / additional force equivalent to the weight of the axis. In error-free operation, the brake can produce the necessary braking torque/braking force and keep the axis at a virtual standstill. When an error occurs, the actual position value exits the parameterizable monitoring window. In this instance, the position controller prevents the axis from sagging and negatively acknowledges the mechanical brake test.

The necessary parameterization of NC and Drive is described in:
References:
Functions Manual Safety Integrated

The brake test must always be started when the axis is at a standstill. For the entire duration of the brake test, the enable signals of the parameterized axis must be set to enable (e.g. the controller inhibit, feed enable signals). Furthermore, the signal at the axis/spindle DB31, ...DBX28.7 (PLC-controlled axis) is to be set to status 1 by the user program for the complete duration of the test.

Before activating the signal DB31, ...DBX28.7 (PLC-controlled axis) the axis is to be switched as "neutral axis", e.g. DB31, ...DBX8.0 - 8.3 (assign NC axis to channel) is to be set to channel 0 as well as DB31, ...DBX8.4 (activation signal when using this byte) is to be set.

The return message:

- about the current status can be queried in DB31, ... DBB68.
- the Nc via the signal DB31, ...DBX63.1 (PLC controls axis) is to be awaited before the block is started. The direction in which the drive must produce its torque/force is specified by the PLC in the form of a "traversing motion" (e.g., via FC 18).

The axis must be able to reach the destination of this movement without risk of collision if the brake is unable to produce the necessary torque/force.

Note

Instructions for FC 18

If FC18 is called for the same axis in the remainder of the user program, the calls must be mutually interlocked. For example, this can be achieved via a common call of this function with an interlocked common data interface for the FC 18 parameters. A second option is to call the FC18 repeatedly, in which case the inactive FC18 will not be processed by the program. A multiple-use interlock must be provided.

The brake test is divided into the following steps:

Brake test sequence		
Step	**Expected feedback**	**Monitoring time value**
Start brake test	DBX 71.0 = 1	TV_BTactiv
Close brake	Bclosed = 1	TV_Bclose
Output traversing command	DBX 64.6 Or DBX 64.7	TV_FeedCommand
Issue test travel command	DBX 62.5 = 1	TV_FXSreached
Wait for the holding time	DBX 62.5 = 1	TV_FXShold
Deselect brake test / open brake	DBX 71.0 = 0	TV_BTactiv
Output test ok		

Declaration of the function

```
Function_BLOCK FB 11
VAR_INPUT
    Start               : BOOL ;        //Start of brake test
    Ackn                : BOOL ;        //Acknowledge error
    Bclosed             : BOOL ;        //Brake closed input (single channel - PLC)
    Axis                : INT;          //Testing axis no.
    TimerNo             : TIMER ;       //Timer from user
    TV_BTactiv          : S5TIME ;      //TimeValue -> brake test active
    TV_Bclose           : S5TIM;        //TimeValue -> close brake
    TV_FeedCommand      : S5TIME ;      //TimeValue -> force FeedCommand
    TV_FXSreached       : S5TIME ;      //TimeValue -> fixed stop reached
    TV_FXShold          : S5TIME ;      //TimeValue -> test brake
END_VAR
VAR_OUTPUT
    CloseBrake     : BOOL ;             //Signal close brake
    MoveAxis       : BOOL ;             //Do move axis
    Done           : BOOL ;
    Error          : BOOL ;
    State          : BYTE ;             //Error byte
END_VAR
```

Description of formal parameters

The following table lists all of the formal parameters of the brake test function

Formal parameters of brake test function			
Signal	**Type**	**Type**	**Remark**
Start	I	BOOL	Starts the brake test
Ack	I	BOOL	Acknowledge fault
Bclosed	I	BOOL	Checkback input whether Close Brake is activated (singlechannel - PLC)
Axis	I	INT	**Axis number of axis to be tested**
TimerNo	I	TIMER	Timer from user program
TV_BTactiv	I	S5TIME	Monitoring time value → brake test active, checking of axis signal DBX 71.0
TV_Bclose	I	S5TIME	Monitoring time value → close brake Check of input signal Bclosed after output CloseBrake has been set.
TV_FeedCommand	I	S5TIME	Monitoring time value → Travel command given Check travel command after MoveAxis has been set
TV_FXSreched	I	S5TIME	Monitoring time value → fixed stop reached
TV_FXShold	I	S5TIME	Monitoring time value → test brake
CloseBrake	A	BOOL	Request, close brake
MoveAxis	A	BOOL	Request, initiate traversing motion
Done	A	BOOL	Test successfully completed
Error	A	BOOL	Error has occurred
State	A	BYTE	Error status

Fault IDs

State	Significance
0	No error
1	Start conditions not fulfilled, e.g. the axis is not in closed-loop control / brake closed / axis inhibited
2	No NC checkback in "Brake test active" signal on selection of brake test
3	No "Brake applied" checkback by input signal Bclosed
4	No travel command output (e.g., axis motion has not been started)
5	Fixed end stop will not be reached → axis RESET was initiated.
6	Traversing inhibit/Approach too slow → fixed stop cannot be reached. TV FXSreached monitoring timeout
7	Brake is not holding at all (the end position is reached)/approach speed is too high
8	Brake opens during the holding time
9	Error when deselecting the brake test
10	Internal error
11	"PLC-controlled axis" signal not enabled in the user program

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 multi-instance-capable.

Example of a call for FB 11:

```
         AN      M          111.1;      //Request to close brake, Z axis of FB
         =        A         85.0;       //Brake control, Z axis
         OPEN               Axis3";     //Brake test, Z axis
         O       I          73.0;       //Brake test trigger, Z axis
         O       M          110.7;      //Brake test running
         FP      M          110.0;
         UN      M          111.4;      //Error has occurred
         S       M          110.7;      //Brake test running
         S       M          110.6;      //Next step
         JCN     m001
         L       DBB        68;
         AW      W#16#F;
         T       MB         115;        //flag channel state
         L       B#16#10
         T       DBB        8;          //Request neutral axis

m001:    U       DBX        68.6;       //Checkback signal, axis is neutral
         U       M          110.6;
         FP      M          110.1;
         R       M          110.6;
         S       M          110.5;      //Next step
         S       DBX        28.7;       //Request PLC-monitored axis

         U       DBX        63.1;       //Checkback signal, axis monitored by PLC
         U       M          110.5;
         FP      M          110.2;
         R       M          110.5;
         S       M          111.0;      //Start brake test for FB
```

```
CALL FB 11, DB 211 (//Brake test block
       Start                  :=M     111.0,      //Start brake test
       Ackn                   := I    3.7,        //Acknowledge error with RESET key
       Bclosed                := I    54.0,       //Return message close brakes
                                                  //controlled
       Axis                   := 3,               //Axis number of axis to be tested
                                                  //Z axis
       TimerNo                := T    110,        //Timer number
       TV_BTactiv             := S5T#200MS,       //Monitoring time value:
                                                  //Brake test active DBX71.0
       TV_Bclose              := S5T#1S,          //Monitoring time value:
                                                  //Brake closed
       TV_FeedCommand         := S5T#1S,          //Monitoring time value:
                                                  //Traversing command output
       TV_FXSreache           := S5T#1S,          //Monitoring time value:
                                                  //Fixed stop reached
       TV_FXShold             := S5T#2S,          //Monitoring time value:
                                                  //Brake test time
       CloseBrake             :=M     111.1,      //Request to close brake
       MoveAxis               :=M     111.2,      Initiate //Request traversing motion //
       Done                   :=M     111.3,      //Test successfully completed
       Error                  :=M     111.4,      //Error has occurred
       State                  := MB   112);       //Error status

OPEN              "Axis3";    //Brake test, Z axis

U      M          111.2;      //Moveaxis
FP     M          111.5;      //FC18 Start
S      M          111.7;      //Start FC18

O      M          111.3;      //Test successfully completed
O      M          111.4;      //Error has occurred
FP     M          110.3;
R      DBX        28.7;       //Request, PLC-monitored axis

UN     DBX        63.1;       //Checkback signal, axis monitored by PLC
U      M          111.0;      //Start brake test for FB
U      M          110.7;      //Brake test running
FP     M          110.4;
R      M          111.0;      //Start brake test for FB
R      M          110.7;      //Brake test running
```

```
//optional begin
         JCN     m002:
         L       MB          115;        //old channel status
         OW      W#16#10;
         T       DBB         8;          //Request channel axis
m002:    NOP     0;
//optional end

         CALL "SpinCtrl" (//Traverse Z axis
                 Start       :=M         111.2,        //Start traversing motion
                 Stop        := FALSE,
                 Funct       := B#16#5,                //Mode: Axis mode
                 Mode        := B#16#1,                //Procedure: Incremental
                 AxisNo      := 3,                     //Axis number of axis to be traversed
                                                       //axis Z-axis
                 Pos         := -5.000000e+000,        //Traversing distance: Minus 5 mm
                 FRate       := 1.000000e+003,         //Feedrate: 1000 mm/min
                 InPos       :=M         113.0,        //Position reached
                 Error       :=M         113.1,        //Error has occurred
                 State       := MB       114);         //Error status

         OPEN                "Axis3";    //Brake test, Z axis
         U       M           113.0;      //Position reached
         O       M           113.1;      //Error has occurred
         FP      M           113.2;
         R       M           111.7;      //Start FC18
```

12.15.10 FB 29: Signal recorder and data trigger diagnostics

Function

Signal recorder

The diagnostics FB allows various diagnostic routines to be performed on the PLC user program. A diagnostic routine logs signal states and signal changes. In this diagnostic routine, function number 1 is assigned to the "Func" parameter. Up to 8 signals of the parameters "Signal_1" to "Signal_8" are recorded in a ring buffer each time one of the signals changes. The current information of parameters "Var1" as BYTE value, and "Var2" and "Var3" as INTEGER values are also stored in the ring buffer.

The number of past OB 1 cycles is also stored in the buffer as additional information. This information enables the graphical evaluation of signals and values in OB 1 cycle grid. The first time the diagnostics FB is called in a new PLC cycle, the "NewCycle" parameter must be set to TRUE. If the diagnostics FB is called several times in the same OB 1 cycle, the "NewCycle" parameter must be set to FALSE for the second and subsequent calls. This prevents a new number of OB 1 cycles from being calculated.

The ring buffer, specified by the user, must have an ARRAY structure specified as in the source code. The array can have any number of elements. A size of 250 elements is recommended. The "ClearBuf" parameter is used to clear the ring buffer and set the BufAddr pointer (I/O parameter) to the start. The instance DB related to the FB is a DB from the user area and is to be transferred to the FB Diagnostics with the parameter "BufDB".

Data trigger

The data trigger function is intended to allow triggering on specific values (or bits) at any permissible memory cell. The cell to be triggered is "rounded" with a bit mask ("AndMask" parameter) before the "TestVal" parameter is compared in the diagnostic block.

Note

The source code for the function is available in the source container of the basic-program library under the name "Diagnose.awl". The instance DB and the ring buffer DB are also defined in this source block. The function call is also described in the function. The DB numbers and the call must be modified.

Declaration of the function

```
FUNCTION_BLOCK FB 29
VAR_INPUT
Func                : INT ;          //Function number: 0 = No function,
                                     //1 = Signal recorder, 2 = Data trigger
    Signal_1        : BOOL ;         //Start of brake test
    Signal_2        : BOOL ;
    Signal_3        : BOOL ;
    Signal_4        : BOOL ;
    Signal_5        : BOOL ;
    Signal_6        : BOOL ;
    Signal_7        : BOOL ;
    Signal_8        : BOOL ;
    NewCycle        : BOOL ;
    Var1            : BYTE ;
    Var2            : INT;
    Var3            : INT;
    BufDB           : INT;
    ClearBuf        : BOOL ;
    DataAdr         : POINTER;       //Area pointer to testing word
    TestVal         : WORD ;         //Value for triggering
    AndMask         : WORD ;         //AND mask to the testing word
END_VAR
VAR_OUTPUT
    TestIsTrue      : BOOL ;
END_VAR
VAR_IN_OUT
    BufAddr         : INT;
END_VAR
```

Structure for ring buffer

```
TITLE =
                                                //Ring buffer DB for FB 29
VERSION : 1.0

STRUCT
    Field: ARRAY [0 .. 249 ] OF STRUCT          //can be any size of this struct

    Cycle : INT ;                               //Delta cycle to last storage in buffer
    Signal_1 : BOOL ;                           //Signal names same as FB 29
    Signal_2 : BOOL ;
    Signal_3 : BOOL ;
    Signal_4 : BOOL ;
    Signal_5 : BOOL ;
    Signal_6 : BOOL ;
    Signal_7 : BOOL ;
    Signal_8 : BOOL ;
    Var1 : BYTE ;
    Var2 : WORD ;
    Var3 : WORD ;
    END_STRUCT;
END_STRUCT;
BEGIN
END_DATA_BLOCK
```

Description of formal parameters

The table below lists all formal parameters of the Diagnostics function:

Formal parameters of diagnostics function					
Signal	**Type**	**Type**	**Value range**	**Remark**	
Func	I	INT	0, 1, 2	Function	
				0:	Switch off
				1:	Signal recorder
				2:	Data trigger
Parameters for function 1					
Signal_1 to Signal_8	I	BOOL		Bit signals checked for change	
NewCycle	I	BOOL		See the "Signal recorder" description above	
Var1	I	BYTE		Additional value	
Var2	I	INT		Additional value	
Var3	I	INT		Additional value	
BufDB	I	INT		Ring buffer DB no.	
ClearBuf	I	BOOL		Delete ring buffer DB and reset pointer BufAddr	
BufAddr	I/O	INT		Target area for read data	
Parameters for function 2					
DataAdr	I	POINTER		Pointer to word to be tested	
TestVal	I	WORD		Comparison value	
AndMask	I	WORD		See description	
TestIsTrue	A	BOOL		Result of comparison	

Configuration steps

- Select function of diagnostics block.
- Define suitable data for the recording as signal recorder or data triggering.
- Find a suitable point or points in the user program for calling the diagnostics FB.
- Create a data block for the ring buffer, see call example.
- Call the diagnostics FB with parameters in the user program.

In function 1, it is advisable to clear the ring buffer with the "ClearBuf" parameter. When the recording phase with function 1 is completed, read out the ring buffer DB in STEP7 with the function "opening the data block in the data view". The content of the ring buffer DB can now be analyzed.

Call example

```
FUNCTION FC 99: VOID
TITLE =
VERSION : 0.0

BEGIN
NETWORK
TITLE = NETWORK

CALL FB 29, DB 80 (
Func                    := 1,
          Signal_1      :=M           100.0,
          Signal_2      :=M           100.1,
          Signal_3      :=M           100.2,
          Signal_4      :=M           100.3,
          Signal_5      :=M           10.4,
          Signal_6      :=M           100.5,
          Signal_7      :=M           100.6,
          Signal_8      :=M           100.7,
          NewCycle      := TRUE,
          Var1          := MB         100,
          BufDB         := 81,
          ClearBuf      :=M           50.0);
END_FUNCTION
```

12.15.11 FC 2: GP_HP Basic program, cyclic section

Function

The complete processing of the NCKPLC interface is carried out in cyclic mode. In order to minimize the execution time of the basic program, only the control/status signals are transmitted cyclically; transfer of the auxiliary functions and G functions only takes place when requested by the NCK.

Declaration

```
FUNCTION FC 2: VOID
// no parameters
```

Call example

As far as the time is concerned, the basic program must be executed **before** the user program. It is, therefore, called first in OB 1.

The following example contains the standard declarations for OB 1 and the calls for the basic program (FC2), the transfer of the MCP signals (FC19), and the acquisition of error and operating messages (FC10).

```
ORGANIZATION_BLOCK OB 1
VAR_TEMP
    OB1_EV_CLASS :          BYTE ;
    OB1_SCAN_1 :            BYTE ;
    OB1_PRIORITY :          BYTE ;
    OB1_OB_NUMBR :          BYTE ;
    OB1_RESERVED_1 :        BYTE ;
    OB1_RESERVED_2 :        BYTE ;
    OB1_PREV_CYCLE :        INT;
    OB1_MIN_CYCLE :         INT;
    OB1_MAX_CYCLE :         INT;
    OB1_DATE_TIME :         DATE_AND_TIME;
END_VAR
BEGIN
CALL FC 2;                          //Call basic program as first FC
//INSERT USER PROGRAM HERE
CALL FC 19 (                          //MCP signals to interface
BAGNo :                               B#16#1,       //Mode group no. 1
ChanNo :                              B#16#1,       //Channel no. 1
SpindleIFNo :                         B#16#4,       //Spindle interface number = 4
FeedHold :                            m22.0,        //Feed stop signal
                                                    //modal
SpindleHold :       db2.dbx151.0);                  //Spindle stop modal
                                                    //in message DB
CALL FC 10 (                                        //Error and operational messages
                    ToUserIF :      TRUE,           //Signals transferred from DB2 to interface
                                                    //to interface
                    Ack :           I6.1);          //Acknowledgment of error messages
                                                    //via I 6.1
END_ORGANIZATION_BLOCK
```

12.15.12 FC 3: GP_PRAL Basic program, interruptdriven section

Function

Blocksynchronized transfers from the NCK to the PLC (auxiliary and G functions) are carried out in the interruptdriven part of the basic program. **Auxiliary functions** are subdivided into normal and highspeed auxiliary functions.

The highspeed functions of an NC block are buffered and the transfer acknowledged to the NC. These are passed to the application interface at the start of the next OB1 cycle.

Highspeed auxiliary functions programmed immediately one after the other, are not lost for the user program. This is ensured by a mechanism in the basic program.

Normal auxiliary functions are acknowledged to the NC only after one completed cycle duration. This allows the application to issue a read disable to the NC.

The **G Functions** are evaluated immediately and passed to the application interface.

NC process interrupts

If the interrupt is triggered by the NC (possible in each IPO cycle), a bit in the local data of OB 40 ("GP_IRFromNCK") is set by the basic program, when the FB 1 parameter "UserIR": = TRUE"). This data is not set on other events (process interrupts through I/Os). This information makes it possible to branch into the associated interrupt routine in the user program in order to initiate the necessary action.

To be able to implement highspeed, jobdriven processing of the user program for the machine, the following NC functions are available in the interrupt processing routine (OB 40 program section) for the PLC user program:

- Selected **auxiliary functions**
- **Tool-change function** for tool-management option
- **Position reached** for positioning axes, indexing axes and spindles with activation via PLC

The functions listed above can or must be evaluated by the user program in OB 40 in order to initiate reactions on the machine. As an example, the revolver switching mechanism can be activated when a T command is programmed on a turning machine.

For further details on programming hardware interrupts (time delay, interruptibility, etc.) refer to the corresponding SIMATIC documentation.

Auxiliary functions

Generally, high-speed or acknowledging auxiliary functions are processed with or without interrupt control independently of any assignment.

Basic-program parameters in FB 1 can be set to define which auxiliary functions (T, H, DL) must be processed solely on an interruptdriven basis by the user program.

Functions which are not assigned via interrupts are only made available by the cyclic basic program as in earlier versions. The change signals of these functions are available in a PLC cycle.

Even if the selection for the auxiliary function groups (T, H, DL) is made using interrupt control, only one interrupt can be processed by the user program for the selected functions.

A bit is set channelspecifically in the local data "GP_AuxFunction" for the user program (if "GP_AuxFunction[1]" is set, then an auxiliary function is available for the 1st channel).

In the related channel-DB the change signal and the function value are available for the user. Das Änderungssignal dieser interrupt driven function is reset to zero in the cyclic basic program section after the execution of at least one full OB1 cycle (max. approx. two OB1 cycles).

Tool change

With the tool-management option, the tool-change command for revolver and the tool change in the spindle is supported by an interrupt. The local data bit "GP_TM" in OB 40 is set for this purpose. The PLC user program can thus check the tool management DB (DB 72 or DB 73) for the tool change function and initiate the tool change operation.

Position reached

In the bit structure, "GP_InPosition" of the local data of OB 40 is specific to the machine axis (each bit corresponds to an axis/spindle, e.g. GP_InPosition[5] corresponds to axis 5).

If a function has been activated via FC 18 (spindle control, positioning axis, indexing axis) for an axis or spindle, the associated "GP_InPosition" bit can be used to implement instantaneous evaluation of the "InPos" signal of the FCs listed above. This feature can be used, for example, to obtain immediate activation of clamps for an indexing axis.

Declaration

```
FUNCTION FC 3 : VOID
// no parameters
```

Call example

As far as the time is concerned, the basic program must be executed **before** other interrupt-driven user programs. It is, therefore, called first in OB 40.

The following example contains the standard declarations for OB 40 and the call for the basic program.

```
ORGANIZATION_BLOCK OB 40
VAR_TEMP
    OB40_EV_CLASS :             BYTE ;
    OB40_STRT_INF :             BYTE ;
    OB40_PRIORITY :             BYTE ;
    OB40_OB_NUMBR :             BYTE ;
    OB40_RESERVED_1 :           BYTE ;
    OB40_MDL_ID :               BYTE ;
    OB40_MDL_ADDR :             INT;
    OB40_POINT_ADDR :           DWORD;
    OB40_DATE_TIME :            DATE_AND_TIME;

//Assigned to basic program
GP_IRFromNCK : BOOL ;                               //Interrupt by NCK for user
GP_TM : BOOL ;                                      //Tool management
GP_InPosition : ARRAY [1..3] OF BOOL;               //Axis-oriented for positioning,
                                                    //Indexing axes, spindles
GP_AuxFunction : ARRAY [1..10] OF BOOL;             //Channel-oriented for auxiliary functions
GP_FMBlock : ARRAY [1..10] OF BOOL;                 //Currently not used
//Further local user data may be defined from this point onwards
END_VAR
BEGIN
    CALL FC 3;
    //INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

12.15.13 FC 5: GP_DIAG Basic program, diagnostic alarm, and module failure

Function

Module defects and module failures are detected in this section of the basic program.

The FC5 block parameter can be used to define whether the PLC is to be placed in Stop mode. The PLC is placed in STOP mode only for incoming events. Exceptions of the parameter "PLC-Stop" are the Profibus-MCPs parameterized at FB 1 (must be connected to the DP1 Bus).

Declaration

```
FUNCTION FC 5: VOID
   VAR_INPUT
       PlcStop: BOOL:= TRUE;
   END_VAR
```

Call example

As far as timing is concerned, the basic program can be executed after other user programs. This is advisable since the FC5 circuitry will place the PLC in Stop mode.

This example contains the standard declarations for OB 82 and OB 86 and the call of the basic program block.

```
ORGANIZATION_BLOCK OB 82
VAR_TEMP
     OB82_EV_CLASS : BYTE ;
     OB82_FLT_ID : BYTE ;
     OB82_PRIORITY : BYTE ;
     OB82_OB_NUMBR : BYTE ;
     OB82_RESERVED_1 : BYTE ;
     OB82_IO_FLAG : BYTE ;
     OB82_MDL_ADDR : INT ;
     OB82_MDL_DEFECT : BOOL ;
     OB82_INT_FAULT : BOOL ;
     OB82_EXT_FAULT : BOOL ;
     OB82_PNT_INFO : BOOL ;
     OB82_EXT_VOLTAGE : BOOL ;
     OB82_FLD_CONNCTR : BOOL ;
     OB82_NO_CONFIG : BOOL ;
     OB82_CONFIG_ERR : BOOL ;
     OB82_MDL_TYPE : BYTE ;
     OB82_SUB_NDL_ERR : BOOL ;
     OB82_COMM_FAULT : BOOL ;
     OB82_MDL_STOP : BOOL ;
     OB82_WTCH_DOG_FLT : BOOL ;
     OB82_INT_PS_FLT : BOOL ;
```

```
     OB82_PRIM_BATT_FLT : BOOL ;
     OB82_BCKUP_BATT_FLT : BOOL ;
     OB82_RESERVED_2 : BOOL ;
     OB82_RACK_FLT : BOOL ;
     OB82_PROC_FLT : BOOL ;
     OB82_EPROM_FLT : BOOL ;
     OB82_RAM_FLT : BOOL ;
     OB82_ADU_FLT : BOOL ;
     OB82_FUSE_FLT : BOOL ;
     OB82_HW_INTR_FLT : BOOL ;
     OB82_RESERVED_3 : BOOL ;
     OB82_DATE_TIME : DATE_AND_TIME;
END_VAR
     BEGIN
        CALL FC 5
           (PlcStop := FALSE) ;
END_ORGANIZATION_BLOCK

ORGANIZATION_BLOCK OB 86
VAR_TEMP
     OB86_EV_CLASS : BYTE ;
     OB86_FLT_ID : BYTE ;
     OB86_PRIORITY : BYTE ;
     OB86_OB_NUMBR : BYTE ;
     OB86_RESERVED_1 : BYTE ;
     OB86_RESERVED_2 : BYTE ;
     OB86_MDL_ADDR : WORD ;
     OB86_RACKS_FLTD : ARRAY [0 .. 31]OF BOOL;
     OB86_DATE_TIME : DATE_AND_TIME;
END_VAR
     BEGIN
        CALL FC 5
           (PlcStop := TRUE) ;
END_ORGANIZATION_BLOCK
```

12.15.14 FC 6: TM_TRANS2 transfer block for tool management and multitool

Function

The TM_TRANS2 block is used for position changes of the tool, status changes and multitool.

The FC 6 block includes the same functionality as for the FC 8, only that in addition the multitool functionality is also integrated. This is the reason that only the parts of the multitool functionality are explained in this chapter. The functionality that is included that goes beyond this is described in "FC 8: TM_TRANS transfer block for tool management (Page 1097)".

Declaration of the function

STL Representation

```
FUNCTION FC 6 : VOID
VAR_INPUT
   Start:              BOOL;
   TaskIdent:          BYTE ;
   TaskIdentNo:        BYTE ;
   NewToolMag:         INT;
   NewToolLoc:         INT;
   OldToolMag:         INT;
   OldToolLoc:         INT;
   Status:             INT;
   MtoolPlaceNum:      INT;
END_VAR
VAR_OUTPUT
   Ready               BOOL;
   Error:              INT;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

The table below lists all formal parameters of the TM_TRANS2 function. Parameters shown in bold differ from FC 8.

Signal	Type	Type	Value range	Remark
Start	I	BOOL		See description of block FC 8
TaskIdent	I	BYTE		See description of block FC 8
TaskIdentNo	I	BYTE		See description of block FC 8
NewToolMag	I	INT		See description of block FC 8
NewToolLoc	I	INT		See description of block FC 8
OldToolMag	I	INT		See description of block FC 8
OldToolLoc	I	INT		See description of block FC 8
Status	I	INT		See description of block FC 8
MtoolPlaceNum	**I**	**INT**		**Multitool location No.**
Ready	O	BOOL		See description of block FC 8
Error	O	INT		See description of block FC 8

12.15.15 FC 7: TM_REV Transfer block for tool change with revolver

Function

After a revolver has been changed, the user will call this block FC TM_REV. The revolver number (corresponding to interface number in DB 73) must be specified in parameter "ChgdRevNo" for this purpose. As this block is called, the associated "Interface active" bit in data block DB 73, word 0 of FC 7 is reset after parameter "Ready" := TRUE is returned.

Block FC TM_REV may be started (with "Start" parameter = "TRUE") only if an activation signal for the appropriate interface (DB 73, word 0) for this transfer has been supplied by the tool management function.

When this job is executed correctly, the output parameter "Ready" contains the value TRUE. The user must then set the"Start" parameter to FALSE or not call the block again.

If the "Ready" parameter is set to FALSE, the error code in the"Error" parameter must be interpreted.

If the error code = 0, then this job must be repeated in the next PLC cycle (e.g. "Start" remains set to "TRUE"). This means that the transfer job has not yet been completed (see example FC 7 call and timing diagram).

The "Start" parameter does not need a signal edge for a subsequent job.

⚠WARNING

A cancellation of a transfer (e.g. through an external signal RESET) is not permitted. The "Start" parameter must always retain the 1 signal until the "Ready" and/or "Error" parameters <> 0.

An error code <> 0 indicates incorrect parameterization.

Note

For further details on tool management (also with regard to PLC) refer to the Description of Functions Tool Management. In addition, PI services for tool management via FB 4, FC 8 and FC 22 are available.

Manual revolver switching

If the revolver is rotated in manual operation, neither a tool change nor an offset selection is associated with this operation. From a data perspective, the programmed tool moves to the toolholder and the old tool back to its place in the revolver during a tool selection also for the revolver, even if this is modeled differently by HMI Operate.

The first step is the removal of the tool from the toolholder back to its location in the revolver. To do this, an asynchronous transfer is performed with FC8 (as an alternative to FC6).

The parameter assignment appears as follows:

```
TaskIdent = 4
TaskIdentNo = Channel no.
NewToolMag = Magazine no. of the revolver
NewToolLoc = Original location of the tool
OldToolMag = Magazine no. of the buffer storage (spindle) = 9998
OldToolLoc = Buffer storage no. of the spindle
Status = 1
```

If the revolver is now turned to an arbitrary position at which a tool is located, this tool must be activated. This is done easiest by the new T programming in the part program. However, if this is to be performed, for example, at the end of the revolver switching by the PLC, the PLC must start an ASUB for this purpose. The current revolver position must be transferred to the ASUB. In this way, the tool at this location is determined in the ASUB and is selected (see Jobshop example in the toolbox).

Declaration of the function

STL representation

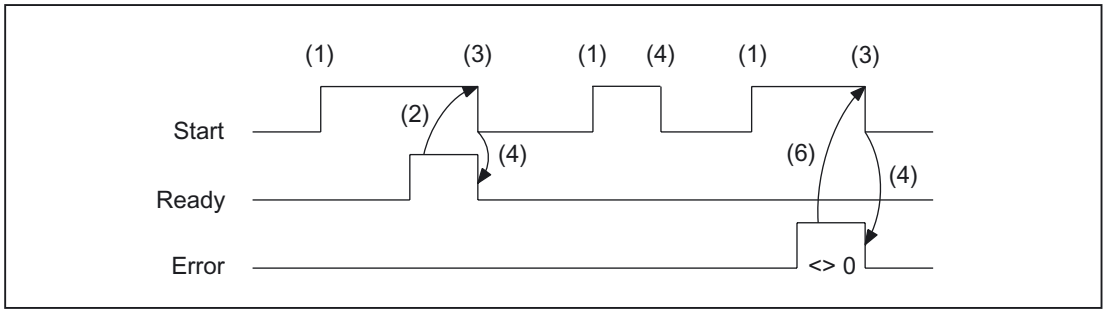
```
FUNCTION FC 7 :      VOID
//NAME :TM_REV
VAR_INPUT
    Start:           BOOL ;
    ChgdRevNo:       BYTE ;
END_VAR
VAR_OUTPUT
    Ready:           BOOL ;
    Error :          INT;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

The table below lists all formal parameters of the TM_REV function.

Signal	Type	Type	Range of values	Remark	
Start	I	BOOL		1 = Start of transfer	
ChgdRevNo	I	BYTE	1...	Number of revolver interface	
Ready	O	BOOL		1 = Transfer complete	
Error	O	INT	0 ... 3	Error checkback	
				0 :	No error has occurred
				1:	No revolver present
				2:	Illegal revolver number in parameter "ChgdRevNo"
				3:	Illegal job ("interface active" signal for selected revolver = "FALSE")

Pulse diagram



(1) Activation of function by means of a positive edge
(2) Positive acknowledgement: Tool management has been transferred
(3) Reset function activation after receipt of acknowledgement
(4) Signal change using FC
(5) This signal chart is not permissible. The job must generally be terminated since the new tool positions must be conveyed to the tool management in the NCK.
(6) Negative acknowledgement: Error has occurred, error code in the output parameter Error

Call example

```
CALL FC 7(                          //Tool-management transfer block
                                    //for Revolver
Start :          m 20.5,            //Start := "1 " => transfer trigger
ChgdRevNo :      DB61.DBB 1,
Ready :          m 20.6,
Error :          DB61.DBW 12);
u m 20.6;                           //Poll ready
r m 20.5;                           //Reset start
spb m001;                           //Jumps, if everything OK
l db61.dbw 12;                      //Error information
ow w#16#0;                          //Evaluate error
JC error;                           //Jumps to troubleshooting, if <> 0
m001:                               // Start of another program
error:
r m 20.5;                           //Reset start, if an error has occurred
```

12.15.16 FC 8: TM_TRANS transfer block for tool management

Function

The user calls this block FC TM-TRANS when the position of the tool or the status of the transfer operation changes. The parameter "TaskIdent" specifies the transfer job for the block FC 8 at the tool management interface:

- For loading/unloading positions
- For spindle change positions
- For revolver change positions as transfer identifier
- Asynchronous transfer
- Asynchronous transfer with location reservation

The interface number is indicated in parameter "TaskIdentNo".

Example for loading point 5:

Parameter "TaskIdent":= 1 and "TaskIdentNo":= 5.

Furthermore, the **current** tool positions and status data (list of "Status" parameter in the following text) are also transferred for this transfer function.

Note

FC8 informs the NCK of the current positions of the old tool.

The NCK knows where the old and the new tool have been located until the position change.

In the case of a transfer without a socalled "old tool" (e.g. on loading), the value 0 is assigned to parameters "OldToolMag", "OldToolLoc".

Block FC TM_TRANS may be started only with "Start" parameter = "TRUE" if an activation signal for the appropriate interface (DB 71, DB 72, DB 73 in word 0) for this transfer has been supplied by the tool management function.

When this job is executed correctly, the output parameter "Ready" contains the value TRUE.

The user must then set the"Start" parameter to FALSE or not call the block again.

If the "Ready" parameter = FALSE, the error code in the "Error" parameter must be interpreted (see Call example FC 8 and timing diagram).

If the error code = 0, then this job must be repeated in the next PLC cycle (e.g. "Start" remains set to "TRUE"). This means that the transfer operation has not yet been completed.

If the user assigns a value of less than 100 to the "Status" parameter, then the associated interface in data block DB 71 or DB 72 or DB 73, word 0 is deactivated (process completed). The appropriate bit for the interface is set to 0 by FC 8.

The "Start" parameter does not need a signal edge for a subsequent job. This means that new parameters can be assigned with "Start = TRUE" immediately when "Ready = TRUE" is received.

Asynchronous transfer

To ensure that changes in the position of a tool are automatically signaled from PLC to tool management (e.g. power failure during an active command or independent changes in the position by the PLC), block FC 8 TM_TRANS with "TaskIdent": = 4 or 5 is called. This call does not require interface activation by tool management.

If parameter "TaskIdent" = 5 the tool management reserves the location in addition to changing the position. The location is only reserved if the tool has been transported from a real magazine to a buffer storage.

A relevant NC channel must be parameterized in the "TaskIdentNo" parameter.

The previous position of the tool is specified in parameters "OldToolMag", "OldToolLoc"; the current position of the tool is specified in parameters "NewToolMag", "NewToolLoc". Status = 1 must be specified.

With status 5, the specified tool remains at location "OldToolMag", "OldToolLoc". This location must be a buffer (e.g. spindle). The real magazine and location must be specified in the parameters "NewToolMag", "NewToolLoc"; the location is at the position of the buffer. This procedure must always be used if the tool management is to be informed of the position of a specific magazine location. This procedure is used for alignment in search strategies.

Note

A cancellation of a transfer (e.g. through an external signal RESET) is not permitted. The "Start" parameter must always retain the 1 signal until the "Ready" and/or "Error" parameters <> 0.

An error code <> 0 indicates incorrect parameterization.

Note

For further details on tool management (also with regard to the PLC) refer to the Function Manual Tool Management. In addition, PI services for tool management via FB 4, FC 7 and FC 22 are available.

Declaration of the function

STL representation

```
FUNCTION FC 8: VOID
//NAME :TM_TRANS
VAR_INPUT
    Start:                BOOL;
    TaskIdent:            BYTE ;
    TaskIdentNo:          BYTE ;
    NewToolMag:           INT;
    NewToolLoc:           INT;
    OldToolMag:           INT;
    OldToolLoc:           INT;
    Status:               INT;
END_VAR
VAR_OUTPUT
    Ready                 BOOL;
    Error :               INT;
END_VAR
BEGIN
END_FUNCTION
```

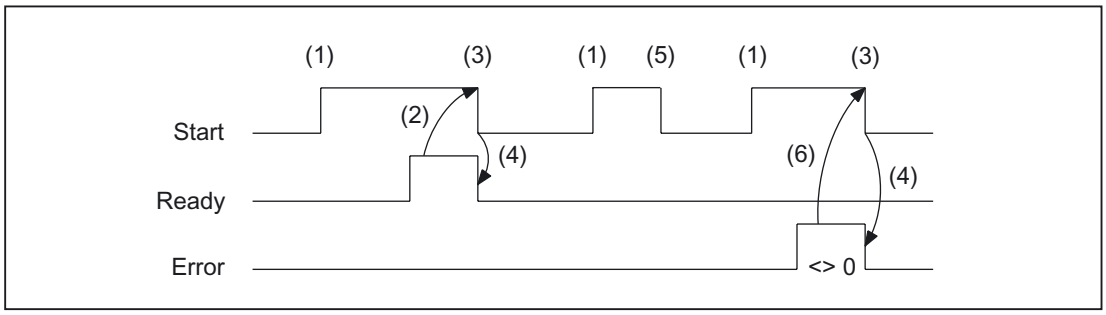
Description of formal parameters

The table below lists all formal parameters of the TM_TRANS function:

Signal	Type	Type	Range of values	Remark	
Start	I	BOOL		1 = Start of transfer	
TaskIdent	I	BYTE	1 ... 5	Interface or task identifier	
				1:	Loading/unloading location
				2:	Spindle change position
				3:	Revolver change position
				4:	Asynchronous transfer
				5:	Asynchronous transfer with location reservation
TaskIdentNo	I	BYTE	1 ...	Number of associated interface or channel number. The upper nibble can specify the interface number for asynchronous transfer (e.g. B#16#12, 1st interface, 2nd channel).	
NewToolMag	I	INT	1, 0 ...	Current magazine number of tool to be replaced	
				-1:	Tool remains at its location NewToolLoc = any value Only permissible if TaskIdent = 2

Signal	Type	Type	Range of values	Remark	
NewToolLoc	I	INT	0 ... max. location number	Current location number of new tool	
OldToolMag	I	INT	-1, 0 ...	Current magazine number of tool to be replaced	
				-1:	Tool remains at its location OldToolLoc = any value. Only permissible if TaskIdent = 2
OldToolLoc	I	INT	Max. location number	Current location number of tool to be replaced	
Status	I	INT	1 ... 7, 103 ... 105	Status information about transfer operation	
Ready	O	BOOL		1= transfer completed	
Error	O	INT	0 ... 65535	Error checkback	
				0:	No error has occurred
				1:	Unknown "TaskIdent"
				2:	Unknown "TaskIdentNo"
				3:	Illegal task ("signal "Interface (SS) active" of selected revolver = "FALSE")
				Other values:	The number corresponds to the error message of the tool management function in the NCK caused by this transfer.

Pulse diagram



(1) Activation of function by means of a positive edge

(2) Positive acknowledgement: Tool management has been transferred

(3) Reset function activation after receipt of acknowledgement

(4) Signal change using FC

(5) This signal chart is not permissible. The job must generally be terminated since the new tool positions must be conveyed to the tool management in the NCK.

(6) Negative acknowledgement: Error occured, error code in the output parameter Error

Status list

Status = 1:
The WZV operation is completed (loading/unloading/reloading, prepare change, change).

The parameters "NewToolMag", "NewToolLoc", "OldToolMag", "OldToolLoc" of the FC 8 block are to be parameterized to the actual positions of the tools involved. Except in the case of preparing change these are normally the specified target position of the tools of the associated WZV interface, see also "Explanations of the formal parameters".

1. In the case of loading/unloading/reloading, the tool has arrived at the required target address. If the bit in the interface in DB 71.DBX (n+0).3 "position at loading point" is enabled, status 1 cannot be used for the function termination. Status 5 must be used for correct termination.
2. In the case of "Prepare change", the new tool is now available. The tool may, for example, be positioned in a buffer (gripper). In some cases, the target (magazine, location) of the old tool has been moved to the toolchange position after placement of the new tool in a buffer. However, the old tool still remains in the spindle. The preparations for a tool change are thus complete. After this acknowledgement, the "Change" command can be received. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag" and "OldToolLoc" correspond to the current tool positions.
3. In the case of "Change" (spindle or revolver), the tools addressed in the interface have now reached the required target addresses.
   The tool-change operation is thus completed.

Status = 2: The "new" tool cannot be made available.

This status is only admissible in conjunction with the "Prepare Change" command. When this status is applied, the PLC must be prevented from making a change with the proposed tool. The proposed (new) tool is disabled by the tool management function in the NCK. A new command is then output by the tool management with a duplo tool. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag", and "OldToolLoc" correspond to the original tool positions.

Status = 3: An error occurred.

The tool positions must not have been changed. Any changes to the magazine positions which have taken place in the meantime must be notified beforehand, for example, with status = 105 via FC 8 transfer block Only then will the tool positions be taken into account by the tool management function.

Status = 4: It would be better to position the "old" tool in the magazine position specified in parameters "OldToolMag" and "OldToolLoc".

This status is permissible only in conjunction with preparation for tool change (change into spindle). After this status has been transferred to the tool management in the NCK, the tool management tries to consider the specified magazine position in the next command. But this is done only when this position is free. Parameters "NewToolMag" and "NewToolLoc" are not taken into account.

Status = 5: The operation is complete.

The "new" tool is **in** the position specified in parameters "NewToolMag", "NewToolLoc". In this case, the specified tool is not really in this position, but is still in the same magazine location. However, this magazine location has been moved to the position set in the parameters (e.g. tool change position). This status may be used only for revolvers, chain-type magazines and disk magazines. Status 5 enables the tool management function to adjust the current position of a magazine and to improve the search strategy for subsequent commands. This status is permissible only in conjunction with loading, unloading, and reloading operations and with preparations for a tool change.

The "OldToolMag" and "OldToolLoc" parameters must be parameterized with the data of a buffer.

- **Loading, reloading**:

  On loading or reloading, a location for the tool is already reserved in the NCK. The machine operator must then insert the tool at the target location. Notice: The location reservation is canceled when the control system is switched on again.

- **Tool-change preparation**:

  Tool motions still to be executed are not carried out until after the tool has been changed.

- **Positioning to load point:**

  If the bit in the interface in DB 71.DBX (n+0).3 "position at loading point" is enabled, status 1 cannot be used for the function termination.

Status = 6: The WZV job has been completed.

This status has the same function as status 1, but, in addition, a reservation of the source location is carried out. This status is only permitted when reloading. The command is ended and the source location of the tool is reserved if the target location is in a buffer magazine.

Status = 7: Initiate repetition of the command "Prepare Tool".

This status is only admissible in conjunction with the "Change tool" command. This status is intended for use when the "new" tool has changed its position (e.g. via an asynchronous command of the "new" tool). After "Ready = 1" has been provided by FC 8, the "Prepare Change" command is repeated automatically with the same tool. For the automatic repetition, a new tool search is carried out. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag", and "OldToolLoc" correspond to the original tool positions.

Status = 103: The "new" tool can be inserted.

This status is permitted only in the tool change preparation, when the PLC may reject the new tool (e.g. in case of MD20310 $MC_TOOL_MANAGEMENT_MASK, bit 4=1 for the possibility, request changed parameter from PLC once again). The tool positions have remained unchanged. This status is thus necessary, when the processing is to be continued in the NCK without an unnecessary stop.

References:
Function Manual Tool Management

Status = 104: The "new" tool is in the position specified in parameters "NewToolMag", "NewToolLoc".

This status is only permissible if the tool is still in the magazine in the same location. The "old" tool is in the position (buffer) specified in parameters "OldToolMag", "OldToolLoc". In this case, however, the new tool is not really in this position, but is still in the same magazine location. However, this magazine location has been moved to the position set in the parameters (e.g. tool change position). This status may be used only in conjunction with revolvers, chaintype magazines and disk magazines for the "Tool change preparation" phase. Status enables the tool management to adjust the current position of a magazine and to improve the search strategy for subsequent commands.

Status = 105: The specified buffer has been reached by all tools involved
(standard case if the operation has not yet been completed).

The tools are in the specified tool positions (parameters "NewToolMag", "NewToolLoc", "OldToolMag", "OldToolLoc").

Status definition

A general rule for the acknowledgement status is that the status information 1 to 7 leads to the termination of the command. If FC 8 receives one of the states, the "Interface active bit" of the interface specified in FC 8 is reset to "0" (see also interface lists DB 71 to DB 73), thus completing the operation. The response if different in the case of status information 103 to 105. When the FC 8 receives one of these items of status information, the “Interface active bit” of this interface remains at “1”. Further processing is required by the user program in the PLC (e.g. continuation of magazine positioning). This item of status information is generally used to transfer changes in position of one or both tools while the operation is still in progress.

Call example

```
CALL FC 8(            //Tool-management transfer block
   Start :=           m 20.5,          //Start := "1 " => transfer trigger
   TaskIdent :=       DB61.DBB 0,
   TaskIdentNo :=     DB61.DBB 1,
   NewToolMag :=      DB61.DBW 2,      //Current position of new tool
   NewToolLoc :=      DB61.DBW 4,
   OldToolMag :=      DB61.DBW 6,      //Current position of old tool
   OldToolLoc :=      DB61.DBW 8,
   Status :=          DB61.DBW 10,     //Status
   Ready :=           m 20.6,
   Error :=           DB61.DBW 12);
u m 20.6;                               //Poll ready
r m 20.5;                               //Reset start
spb m001;                               //Jumps, if everything OK
l DB61.dbw 12;                          //Error information
ow w#16#0;                              //Evaluate error
JC error;                               //Jumps to troubleshooting

m001:                                   //Normal branch

error:                                  //Troubleshooting
r m 20.5:                               //Reset start
```

12.15.17 FC 9: ASUB startup of asynchronous subprograms

Function

The FC ASUB can be used to trigger any functions in the NC. Before an ASUB can be started from the PLC, it must have been selected and parameterized by an NC program or by FB 4 (PI service ASUB). The channel and the interrupt numbers for the parameters in FC 9 must match here.

Once prepared in this way, it can be started at any time from the PLC. The NC program running on the channel in question is interrupted by the asynchronous subprogram. Only one ASUB can be started in the same channel at a time. If the start parameter is set to logical 1 for two FC 9s within **one** PLC cycle, the ASUBs are started in the sequence in which they are called.

The start parameter must be set to logic 0 by the user once the ASUB has been terminated (Done) or if an error has occurred.

For the purpose of job processing, every FC ASUB requires its own WORD parameter "Ref" from the global user memory area. This parameter is for internal use only and must not be changed by the user. The parameter "Ref" is initialized with the value 0 in the first OB 1 cycle and, for this reason, **every FC 9 must be called absolutely**. Alternatively, the user can initialize parameter "Ref" with a value of 0 during startup. This option makes conditional calls possible. When a conditional call is activated by parameter "Start" = 1, it must remain active until parameter "Done" has made the transition from 1 to 0.

Note

The FB 4 call must be terminated before the FC 9 can be started. FC 9 cannot be started if "Emergency stop" is set. Neither can FC 9 be started if the channel reset is active.

Declaration of the function

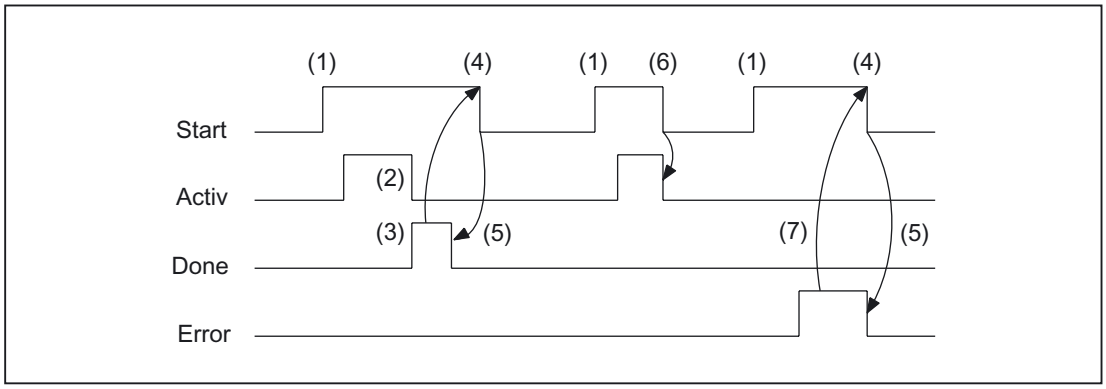
```
FUNCTION FC 9: VOID
//NAME :ASUP
VAR_INPUT
    Start:         BOOL;
    ChanNo:        INT;
    IntNo:         INT;
END_VAR
VAR_OUTPUT
    Active:        BOOL;
    Done :         BOOL;
    Error :        BOOL;
    StartErr:      BOOL;
END_VAR
VAR_IN_OUT
    Ref:           WORD ;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the ASUB function.

Signal	Type	Type	Range of values	Remark
Start	I	BOOL		
ChanNo	I	INT	1 ... 10	No. of the NC channel
IntNo	I	INT	1 ... 8	Interrupt No.
Active	O	BOOL		1 = Active
Done	O	BOOL		1 = ASUB completed
Error	O	BOOL		1 = Interrupt switched off
StartErr	O	BOOL		1 = Interrupt number not assigned or deleted
Ref	I/O	WORD	Global variable (MW, DBW,..)	1 word per FC 9 (for internal use)

Pulse diagram



(1) Activation of function

(2) ASUB is active

(3) Positive acknowledgement: ASUB completed

(4) Reset function activation after receipt of acknowledgement

(5) Signal change using FC

(6) Not permitted If function activation is reset prior to receipt of acknowledgement, the output signals are not updated without the operational sequence of the activated function being affected.

(7) Negative acknowledgement: Error has occurred

Call example

```
CALL FC 9(                          //Start an asynchronous subprogram
                                    //in channel 1 interrupt number 1
     Start :        I 45.7,
     ChanNo :       1,
     IntNo :        1,
     Active :       M 204.0,
     Done :         M204.1,
     Error :        M 204.4,
     StartErr :     M 204.5,
     Ref :          MW 200);
```

12.15.18 FC 10: AL_MSG error and operating messages

Function

FC AL_MSG evaluates the signals entered in DB 2 and displays them as incoming and outgoing error and operational messages on the HMI.

The incoming signals (positive edge) are displayed immediately in the case of both error and operational messages.

Outgoing signals (negative edge) are only canceled immediately in the case of operating messages. Error messages remain stored on the HMI - even if the signals no longer exist - until the "acknowledge" parameter is issued, WCS.e. until the user acknowledges the messages.

The "ToUserIF" parameter can be used to transfer the group signals for the feed, read and NC start disabling signals and feed stop signal to the existing axis, spindle and channel interfaces. The group signals are transferred to the user interface directly from the status information in DB 2 irrespective of an interrupt acknowledgment.

1. If parameter "ToUserIF": is set to FALSE, signals are not transferred to the user interface. In this case, the user must take measures in his PLC program to ensure that these signals are influenced in the interface.
2. If parameter "ToUserIF": is set to TRUE, all signals listed above are sent to the user interface as a group signal in each case. The user PLC program can, therefore, influence these signals only via DB 2 in connection with a message or interrupt output. The appropriate information is overwritten in the user interface.

As an alternative to the procedure described under paragraph 2, the user can influence the disable and stop signals without a message output by applying a disable or stop signal state to the interface signals after FC AL_MSG has been called.

The following program illustrates this method:

```
CALL FC 10 (
   ToUserIF :=       TRUE,
   Ack :=            I 6.1);

u m 50.0;                          //Feed disable for channel 1
to DB 21;
s dbx 6.0;                         //Setting the blocking condition,
                                   //Resetting is done via FC AL_MSG,
                                   //if M 50.0 outputs the signal "0".
```

FB 1-Parameter "ExtendAlMsg"

With the activation of the parameter a new structuring of the DB 2 becomes effective (see "Interface PLC/HMI (Page 933)"). Upon activation the bit fields are available to the user for the disable and halt signals, which do not generate any alarms, messages. As a result, the user need not implement the aforesaid measures. The desired functionality is given automatically by a simple setting, resetting of signals in the new DB 2 areas.

The error and the operating messages are stored by the user in data block DB 2 (see description of DB 2 in the lists of interface signals).

Note

In DB 2, a "1" signal must be present for several OB1 cycles to ensure that a message can also be displayed on the HMI. There is an upper limit for the number of interrupts and messages that can be pending at the same time. This upper limit is dependent on the PLC CPU. On PLC 317-2DP, the upper limit for messages pending simultaneously is 60.

See also Parameter Manual (Lists, Manual 2), chapter on PLC Alarms / Messages

Declaration of the function

STL Representation

```
FUNCTION FC 10 :        VOID
    // NAME:            AL_MSG
VAR_INPUT
    ToUserIF :          BOOL ;
    Ack :               BOOL ;
END_VAR
END_FUNCTION
```

Description of formal parameters

The table below lists all formal parameters of the AL-MSG function.

Signal	Type	Type	Range of values	Remark
ToUserIF	I	BOOL		1 = Transfer the signals to user interface every cycle
Ack	I	BOOL		1 = Acknowledge error messages.

Call example

```
CALL FC 10 (                    //Error and operational messages
   ToUserIF :=     TRUE,        //Signals from DB 2 are transferred to
                                //interface
   Ack :=          I6.1);       //Acknowledgement of the error message done via
                                //Input E6.1.
```

12.15.19 FC 12: AUXFU call interface for user with auxiliary functions

Function

FC AUXFU is generally called on an eventdriven basis in the basic program if the channel transferred in the input parameter contains new auxiliary functions. The PLC user can extend FC AUXFU with program instructions for processing his auxiliary function to avoid cyclic polling of the channel DBs. This mechanism permits auxiliary functions to be processed on a jobdriven basis. FC AUXFU is supplied as a compiled empty block in the basic program. In this case, the basic program supplies parameter "Chan" with the channel number. The PLC user knows which channel has new auxiliary functions available. The new auxiliary functions can be determined by the auxiliary-function change signals in the channel concerned.

Declaration of the function

```
FUNCTION FC 12: VOID            //Event control of auxiliary functions
VAR_INPUT
    Chan:      BYTE ;
END_VAR
BEGIN
    BE;
END_FUNCTION
```

Explanation of formal parameters

The table below lists all formal parameters of the AUXFU function:

Signal	Type	Type	Value range	Remark
Chan	I	BYTE	0 ... 9	No. of NC channel -1

Example

```
FUNCTION FC 12: VOID          //Event control of auxiliary functions
VAR_INPUT
    Chan:        BYTE ;       //Parameter is supplied by basic program
END_VAR
VAR_TEMP
    ChanDB:      INT;
END_VAR

BEGIN
L Chan;                       //Channel index no., (0,1,2,..)
+ 21;                         //Channel DB offset
T ChanDB;                     //Save channel DB no.
TO DB[ChanDB];                //Channel DB is opened indirectly
// Auxiliary-function change signals are now scanned, etc.
    BE;
END_FUNCTION
```

12.15.20 FC 13: BHGDisp Display control for handheld unit

Function

This block carries out the display control for the handheld unit (HHU or HT2). The information to be output on the display is stored as 32 characters in string data ChrArray (these are 64 characters when using an HT2). A fixed text assignment of 32 or 64 characters is, therefore, required for this string when the data block is created.

16 characters are sent to BHG/HT2 for each job, which lasts for several OB 1-cycle. The assignment of the characters in ChrArray to each line is unique. For line 1 the characters 1 to 16 and for the line 2 the characters 17 to 32 of the string data ChrArray are transferred. In addition, for HT2 the line 3 with the characters 33 to 48 and line 4 made of characters 49 to 64 are shown.

The block checks, whether the minimum length of the ChrArray is available for operating the BHG or the HT2. If less characters are present in the string data than are to be displayed, then the line is filled with blanks.

By setting parameter Row to 0, it is possible to suppress the display (e.g. if several variables in one or several PLC cycles need to be entered in the string without any display output). If several lines are to be updated "simultaneously" (transfer of the characters to the lines lasts for several OB 1 cycles) (Parameter Row > 1), then the lines are updated one by one each with 16 characters per line.

Variable portions within the string can be inserted by means of the numerical converter functionality (optional). For the numeric converter the parameter "Convert" must be set to TRUE. The variable to be displayed is referenced via the pointer Addr. Parameter "DataType" contains the format description of this parameter (see parameter table). The number of bytes of the variable is linked to the format description. The address justified to the right within the string is specified by parameter "StringAddr". The number of written characters is shown in the parameter table.

Signals

Byte 1 is supplied by the output signals of the HHU and the character specifications are supplied by the block. These may not be written by the PLC user program.

Additional parameters

HHU

The pointer parameters for the input and output data of the handheld unit must be parameterized in the start OB 100 in FB 1, DB 7. Parameter "BHGIn" corresponds to the input data of the PLC from the handheld unit (data received by PLC). Parameter "BHGOut" corresponds to the output data of the PLC to the handheld unit (data transmitted by PLC). These two pointers must be set to the starting point of the relevant data area (which is also parameterized in SDB 210 with an MPI link).

For operating a BHG a "2" is to be entered at FB 1 parameter BHG.

HT2

For using a HT2 a "5" is to be entered at FB 1 parameter BHG. The pointer parameter of the input and output data are also to be supplied, as described above.

In the parameters BHGRecGDNo and BHGRecGBZNo the value is to be entered, which is configured at the S2 of the DIP-Fix-switch (rotary coding switch) of the connection module of the HT2.

Note

Numerical conversion

If the numerical converter is used to display information, then it is better to avoid performing a conversion in every PLC cycle for the sake of reducing the PLC cycle time.

The conversion routine can be used independently of the display control. This is to be queried in parameter row "0", although the convert parameter should be set. Consequently, only the string is processed, and the converter routine is executed.

High display resolution

If, for example, the actual axis value is to be displayed with a higher resolution, the following should be noted:
Variables will continue to be read with FB 2 or FB 5. REAL 2 is used instead of ANY pointer BYTE 8 as the criterion for output as a 64-bit floating point number (e.g. P#M100.0 REAL 2). When specifying the 64-bit floating point number on the HHU/HT2, an output format of up to 14 places, split freely between places before and after the decimal point, can be selected instead of fixed formats.

Declaration of the function

STL representation

```
DATA_BLOCK "strdat"                  // DB-Number defined in the symbol file

   STRUCT                            //32 characters are defined
     disp:          STRING [32]:= 'character_line1 character_line2';
   END_STRUCT;
BEGIN
END_DATA_BLOCK

FUNCTION FC 13 : VOID
   VAR INPUT
     Row :            BYTE ;         //Display line (see table)
     ChrArray :       STRING ;       //Transfer at least string[32/64]
     Convert :        BOOL;          //Activate numerical conversion
     Addr:            POINTER;       //Points to the variable being converted
     DataType :       BYTE ;         //Data type of the variables
     StringAddr :     INT;           //right-justified string address (1...32/64)
     Digits :         BYTE ;         //Number of decimal places (1...3)
END VAR
VAR OUTPUT
     Error :          BOOL;          //Conversion or string error
END VAR
```

Description of formal parameters

The table below lists all formal parameters of the BHGDisp function:

Signal	Type	Type	Value range	Remark	
Row	I	BYTE	0 ... B#16#F	Display-line "binary" evaluation:	
				0:	no display output
				1:	Line 1
				2:	Line 2
				3:	Line 1 and line 2 to be changed
				4:	Line 3
				5:	Line 1 and line 3 to be changed
				8:	Line 4
				B#16#F	automatic change of all 4 lines

Signal	Type	Type	Value range	Remark	
ChrArray	I	STRING	>= string[32] [DBName].[VarName]	This string contains the entire display content For HT2 the string with 64 characters must be created.	
Convert	I	BOOL		Activation of numerical conversion	
Addr	I	Pointer		Points to the variable to be converted	
DataType	I	BYTE	1 ... 8, B#16#13	Data type of the tag	
				1:	BOOL, 1 character
				2:	BYTE, 3 characters
				3:	CHAR, 1 character
				4:	WORD, 5 characters
				5:	INT, 6 characters
				6:	DWORD, 7 characters
				7:	DINT, 8 characters
				8:	REAL, 9 characters (7 digits plus a sign and a decimal point; for places after the decimal point, refer to the Digits Parameter)
				B#16#13:	String, up to 32/64 characters, Addr must be a pointer to a STRING.
				B#16#30	REAL64, (12 characters: 10 digits plus a sign and a decimal point; for places after the decimal point, refer to the Digits Parameter)
StringAddr	I	INT	1 ... 32/64	Right-justified address within variable ChrArray	
Digits	I	BYTE	for REAL data type: 1 ... 4 for REAL64 data type: 1 ... 9	Number of places after the decimal point:	
Error	O	BOOL		Error: • created chr/array too small, • conversion error, • numerical overflow, • StringAddr faulty	

Ranges of values

Ranges of values of data types	
Data type	**Representable numerical range**
BOOL	0, 1
BYTE	0 ... 255
WORD	0 ... 65535
INT	- 32768 to + 32767
DWORD	0 ... 9999999
DINT	- 9999999 to + 9999999
REAL (Digits := 1)	- 999999.9 to + 999999.9
REAL (Digits := 2)	- 99999.99 to + 99999.99
REAL (Digits := 3)	- 9999.999 to + 9999.999
REAL (Digits := 4)	- 999.9999 to + 999.9999
...	...
REAL (Digits := 9)	- 0.9999999 to + 0.9999999

Call example

//DB with the name strdat is declared in symbol table, data element disp is declared as String[32] (in HT2:
//String[64]) and completely assigned with characters

```
CALL FC 13 (
   Row :              MB 26,
   ChrArray :         "strdat".disp,
   Convert :          M 90.1,
   Addr :             P#M 20.0,          //Number to be converted
   DataType :         MB 28,             //Data type of the variables
   StringAddr :       MW 30,
   Digits :           B#16#3,            //3 decimal places
   Error :            M 90.2);
```

12.15.21 FC 17: YDelta Star-Delta changeover

Function

The block for the star-delta changeover controls the timing of the defined switching logic such that the changeover can be performed in either direction even when the spindle is running. This block may be used only for digital main spindle drives and must be called separately for each spindle.

The changeover operation is implemented via 2 separate contactors in a sequence involving 4 steps:

Step 1:	Deleting the interface signal DB31, ...DBX21.5 (motor selection done) in the related axis-DB and registering the changeover process via A with DB31, ...DBX21.3 (motor selection).
Step 2:	As soon as the checkback message IS DB31, ... DBX93.7 (Pulse enabled) = 0 and the acknowledgment of the announced motor selection from the drive have appeared, the currently energized contactor drops out.
Step 3:	The other contactor is energized after the time period set by the user in parameter "TimeVal" has elapsed.
Step 4:	After a further delay, the changeover is signaled to the drive with NST DB31, ...DBX21.5 (motor selection done):

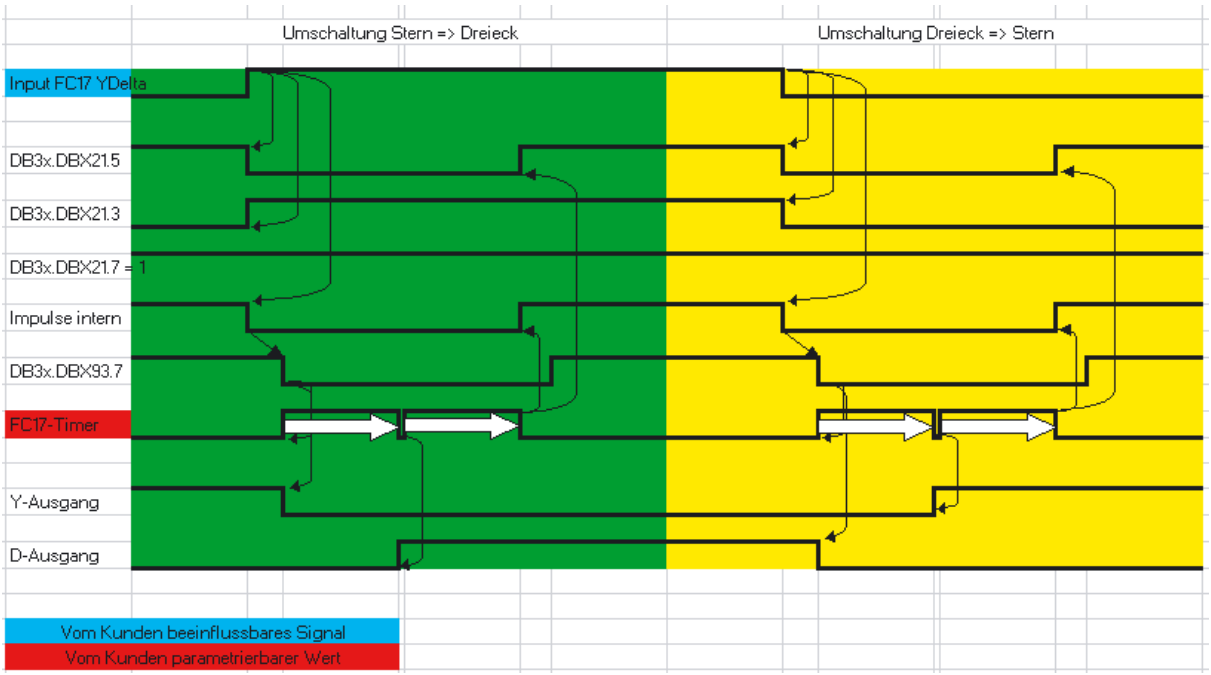


Figure 12-28 Star-delta changeover

For more information on motor speed adjustments see:
References:
Functions Manual Basic Functions; Spindles (S1); Chapter "Configurable gear adjustments"
Functions Manual Basic Functions; Speeds, Reference/Actual value syst., Regulation (G2)

Error message

If the parameter "SpindleIFNo" is not in the permissible range, the PLC is stopped with output of interrupt message number 401702.

Special features

When parameterizing "TimeVal" with the value 0, a default value of 100 ms is used. With a value of less than 50 ms, the minimum setting of 50 ms is applied.

The block must be called unconditionally.

Note

Switchover does not take place if the spindle is in an axis mode such as M70, SPOS.

Boundary conditions

Star-delta changeover on digital main spindle drives initiates a process, which contains closed-loop control sequences. Since the closed-loop control system supports automatic star-delta changeover, certain restrictions should be noted:

- Due to the automatic deactivation of the pulse on the drive, the NST DB31, ...DBX93.7 (Pulse enabled) deactivates simultaneously the IS DB31, ...DBX61.7 (current controller active) and DB31, ...DBX61.6 (speed controller active).
- If a changeover from star to delta takes place while the spindle is rotating and the spindle position controller is switched on, IS "Position controller active" (DB31, ... DBX61.5), this triggers alarm 25050 "Contour monitoring".
- Once the star-delta changeover has been initiated with FC 17, it cannot be delayed by the user, e.g. by waiting until the star/delta contactors change over during the course of operation. The user can implement this signal interaction with PLC logic.

Declaration of the function

STL representation

```
VAR_INPUT
    YDelta:          BOOL;          //Star = 0, delta = 1
    SpindleIFNo:     INT;           //Machine axis number
    TimeVal:         S5TIME ;       //Time value
    TimerNo :        INT;           //User timer for changeover time
END_VAR
VAR_OUTPUT
    Y:               BOOL;          //Star contactor
    Delta:           BOOL;          //Delta contactor
END_VAR
VAR_IN_OUT
    Ref:             WORD ;         //Block status word (instance)
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the YDelta function:

Signal	Type	Type	Value range	Comment
YDelta	I	BOOL		= Star = Triangle The changeover edge of the signal initiates the changeover operation.
SpindleIFNo	I	INT	1 ...	Number of the axis interface declared as a spindle
TimeVal	I	S5time	0 ...	Switchover time
TimerNo	I	INT	10 ...	Timer for programming the wait time
Y	O	BOOL		Energizing of star contactor
Delta	O	BOOL		Energizing of delta contactor
Ref	I/O	WORD		Instance for status information Internal use

Call example

```
CALL FC 17 (
    YDelta :           I 45.7,          //Star delta
    SpindleIFNo :      4,
    TimeVal :          S5T#150ms,
    TimerNo :          10,              //Timer 10
    Y :                Q 52.3,          //Star contactor
    Delta :            Q 52.4,          //Delta contactor
    Ref :              mw 50);          //Instance
```

12.15.22 FC 18: SpinCtrl Spindle control

Function

FC SpinCtrl can be used to control spindles and axes from the PLC.

References
/FB1/Function Manual, Basic Functions; Spindles (S1)
Function Manual, Extended Functions; Positioning Axes (P2)
Function Manual, Extended Functions; Indexing Axes (T1)

This block supports the following functions:

- Position spindle
- Rotate spindle
- Oscillate spindle
- Indexing axes
- Positioning axes

Each function is activated by the positive edge of the appropriate initiation signal (start, stop). This signal must remain in the logic "1" status until the function has been acknowledged positively or negatively by InPos="1" or Error = "1". The output parameters are deleted when the relevant trigger signal is reset and the function terminated.

To be able to control an axis or spindle via the PLC, it must be activated for the PLC. This can, for example, be achieved by calling the FC "SpinCtrl" with activation of the "Start" or "Stop" parameter. In this case, the FC "SpinCtrl" requests control over the spindle/axis from the NC.

The NC feeds back the status of this spindle/axis in byte 68 in the associated spindle/axis interface (DB31, ...) (see interface lists). Once the axis/spindle is operating under PLC control, the travel command for the active status can be evaluated via the relevant axis interface.

On completion ("InPos" is TRUE, "Start" changes to zero), the axis/spindle check function is switched to a neutral status by FC "SpinCtrl".

Alternatively, the PLC user program can also request the check for the PLC prior to calling FC "SpinCtrl".

By calling this function several times in succession, a better response by the spindle/axis can be obtained as the changeover process in the FC can be omitted.

Activation through the PLC user program is executed in the corresponding spindle interface in byte 8.

After return of the check, the spindle can again be programmed by the NC program.

Note

Please note:

FC 18 must be called cyclically until signal "InPos" or, in the case of an error "Error", produces an edge transition of 1 to 0. Only when the "InPos"/"Error" signal has a 0 state can the axis/spindle concerned be "started" or "stopped" again (the next "start" must be delayed by at least one PLC cycle). This also applies when the assignment in data byte 8 on the axial interface has been changed.

Abort:

The function cannot be aborted by means of parameter "Start" or "Stop", but only by means of the axial interface signals (e.g. delete distance-to-go). The axial interface also returns status signals of the axis that may need to be evaluated (e.g. exact stop, traverse command).

InPos on spindle - rotation/oscillation:

For the function "Rotate spindle" and also for "Oscillate spindle" the meaning of the "InPos" parameter is defined as follows:
Setpoint speed is output → function started without errors.
Reaching the desired spindle speed must be evaluated via the spindle interface.

Simultaneity:

Several axes can be traversed simultaneously or subject to a delay by FC 18 blocks. The upper limit is defined by the maximum number of axes. The NCK handles the PLC function request (FC 18) via independent interfaces for each axis/spindle.

Axis disable:

With the axis disabled, an axis controlled via FC 18 will not move. Only a simulated actual value is generated (behavior as with NC programming).

⚠ WARNING

If several block calls (FC 18) have been programmed for the same axis/spindle in the PLC user program, then the functions concerned must be interlocked by conditional calls in the user program. The conditional call of a started block (parameter Start or Stop = TRUE) must be called cyclically until the signal state of output parameter "Active" or "InPos" changes from 1 to 0.

Functions

1. Position spindle:

The following signals are relevant:

Start:	Initiation signal
Funct:	"1" = Position spindle
Mode:	Positioning mode 1, 2, 3, 4
AxisNo:	Number of machine axis
Pos:	Position
FRate:	Positioning speed, if FRate = 0, the value from MD35300 $MA_SPIND_POSCTRL_VELO (position control activation speed) is taken
InPos:	Is set to "1" when position is reached with "Exact stop fine"
Error :	With positioning error = "1"
State :	Error code

2. Rotate spindle:

The following signals are relevant:

Start:	Initiation signal for start rotation
Stop:	Initiation signal for stop rotation
Funct:	"2" = Rotate spindle
Mode:	Positioning mode 5 (direction of rotation M4)
	Positioning mode < >5 (direction of rotation M3)
AxisNo:	Number of machine axis
FRate:	Spindle speed
InPos:	Function has started without an error
Error :	With positioning error = "1"
State :	Error code

3. Oscillate spindle:

The following signals are relevant:

Start:	Initiation signal for start oscillation
Stop:	Initiation signal for stop oscillation
Funct:	"3" = Oscillate spindle
AxisNo:	Number of machine axis
Pos:	Set gear step
InPos:	Setpoint speed is output
Error :	With positioning error = "1"
State :	Error code

The oscillation speed is taken from machine data:
MD35400 $MA_SPIND_OSCILL_DES_VELO

MD35010 $MA_GEAR_STEP_ CHANGE_ENABLE = 0	Function	MD35010 $MA_GEAR_STEP_ CHANGE_ENABLE = 1	Function
Pos = 0	Oscillating	Pos = 0	
Pos = 1	Oscillating	Pos = 1	Oscillation with gear stage change M41
Pos = 2	Oscillating	Pos = 2	Oscillation with gear stage change M42
Pos = 3	Oscillating	Pos = 3	Oscillation with gear stage change M43
Pos = 4	Oscillating	Pos = 4	Oscillation with gear stage change M44
Pos = 5	Oscillating	Pos = 5	Oscillation with gear stage change M45

4. Traverse indexing axes:

The following signals are relevant:

Start: Initiation signal
Funct: "4" = Indexing axis

Note

With **Funct: "4" = Indexing axis**

The modulo conversion can be compared with approaching the indexing position via `POS[AX] = CIC (value)` in the part program.

`Mode:`	`Positioning mode 0, 1, 2, 3, 4`
`AxisNo:`	`Number of machine axis`
`Pos:`	`Indexing position`
`FRate:`	`Positioning speed; if FRate = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).`
`InPos:`	`Is set to "1" when position is reached with "Exact stop fine"`
`Error :`	`With positioning error = "1"`
`State :`	`Error code`

5. to 8. Position axes:

The following signals are relevant:

Start:	Initiation signal
Funct:	"5 to 8" = Position axes
Mode:	Positioning mode 0, 1, 2, 3, 4
AxisNo:	Number of machine axis
Pos:	Position
FRate:	Positioning speed; if FRate = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).
InPos:	Is set to "1" when position is reached with "Exact stop fine"
Error :	With positioning error = "1"
State :	Error code

9. Rotate spindle with automatic gear stage selection:

The following signals are relevant:

Start:	Initiation signal for start rotation
Stop:	Initiation signal for stop rotation
Funct:	"9" = Rotate spindle with gear stage selection
Mode:	Positioning mode 5 (direction of rotation M4)
	Positioning mode < >5 (direction of rotation M3)
AxisNo:	Number of machine axis
FRate:	Spindle speed
InPos:	Setpoint speed is output
Error :	With positioning error = "1"
State :	Error code

10./11. Rotate spindle with constant cutting rate:

The "Constant cutting rate" function must be activated by the NC program in order for this to be executed.

The following signals are relevant:

Start:	Initiation signal for start rotation
Stop:	Initiation signal for stop rotation
Funct:	"B#16#0A = Rotate spindle with constant cutting rate (m/min)
Funct:	"B#16#0B = Rotate spindle with constant cutting rate (feet/min)
Mode:	Positioning mode 5 (direction of rotation M4) Positioning mode < >5 (direction of rotation M3)
AxisNo:	Number of machine axis
FRate:	Cutting rate
InPos:	Setpoint speed is output
Error :	With position error = "1"
State :	Error code

Declaration of the function

```
FUNCTION FC 18: VOID                    //SpinCtrl
VAR_INPUT
    Start:          BOOL;
    Stop:           BOOL;
    Funct:          BYTE ;
    Mode:           BYTE ;
    AxisNo:         INT;
    Pos:            REAL;
    FRate:          REAL;
END_VAR
VAR_OUTPUT
    InPos:          BOOL;
    Error :         BOOL;
    State :         BYTE ;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the SpinCtrl function.

Signal	Type	Type	Range of values	Remark
Start	I	BOOL		Start spindle control from the PLC
Stop	I	BOOL		Stop spindle control from the PLC
Funct	I	BYTE	1 to B#16#0B	1: Position spindle 2: Rotate spindle 3: Oscillate spindle 4: Indexing axis 5: Positioning axis metric 6: Positioning axis inch 7: PosAxis metric with handwheel override 8: PosAxis inch with handwheel override 9: Rotate spindle with automatic gear stage selection A: Rotate spindle with constant cutting rate (m/min) B: Rotate spindle with constant cutting rate (feet/min)
Mode	I	BYTE	0 to 5	0: Pos to absolute pos 1: Pos incrementally 2: Pos shortest path 3: Pos absolute, positive approach direction 4: Pos absolute, negative approach direction 5: Direction of rotation as for M4
AxisNo	I	INT	1 - 31	No. of axis/spindle to be traversed
Pos	I	REAL	∓ 0.1469368 I -38 to ∓ 0.1701412 I +39	Rotary axis: Degrees Indexing axis: Indexing position Linear axis: mm or inches
FRate	I	REAL	∓ 0.1469368 I -38 to ∓ 0.1701412 I +39	Rotary axis and spindle: rev/min See under table containing info about FRate
InPos	O	BOOL		1 = Position reached, or function executed
Error	O	BOOL		1 = Error
State	O	BYTE	0 to 255	Error code

FRate

The feedrate in FC 18 can also be specified as:

- Cutting rate with unit m/min or feet/min
- Constant grinding wheel surface speed in m/s or feet/s

These alternative velocity settings can be used only if this function is activated by the NC program. Checkback signals for successful activation can be found in byte 84 on the axis interface.

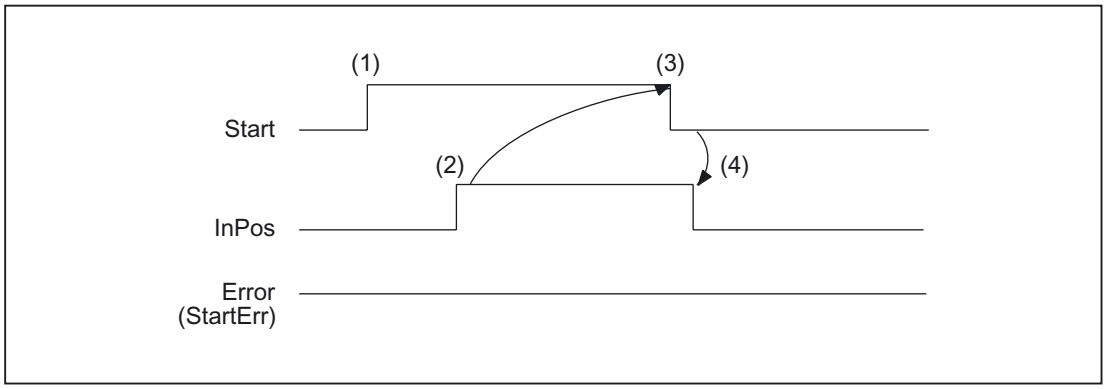
Error identifiers

If a function could not be executed, this is indicated by the "Error" state parameter being set to 'logic 1'. The error cause is coded at block output "State":

State		Meaning
Errors caused by PLC handling:		
1	B#16#1	Several functions of the axis/spindle were activated simultaneously
20	B#16#14	A function was started without the position being reached
30	B#16#1e	The axis/spindle was transferred to the NC while still in motion
40	B#16#28	The axis is programmed by the NC program, NCK internal error
50	B#16#32	Permanently assigned PLC axis Traverses (JOG) or is referencing
60	B#16#3C	Permanently assigned PLC axis Channel status does not permit a start
Errors that occur due to handling of the NCK. The alarm numbers are described in the Diagnostics Manual:		
100	B#16#64	False position programmed for axis/spindle (corresponds to alarm number 16830)
101	B#16#65	Programmed speed is too high
102	B#16#66	Incorrect value range for constant cutting rate (corresponds to alarm number 14840)
104	B#16#68	Following spindle: illegal programming (corresponds to alarm number 22030)
105	B#16#69	No measuring system available (corresponds to alarm number 16770)
106	B#16#6a	Positioning process of the axis still active (corresponds to alarm number 22052)
107	B#16#6b	Reference mark not found (corresponds to alarm number 22051)
108	B#16#6c	No transition from speed control to position control (corresponds to alarm number 22050)
109	B#16#6d	Reference mark not found (corresponds to alarm number 22051)
110	B#16#6e	Velocity/speed is negative
111	B#16#6f	Setpoint speed is zero
112	B#16#70	Invalid gear stage
115	B#16#73	Programmed position has not been reached

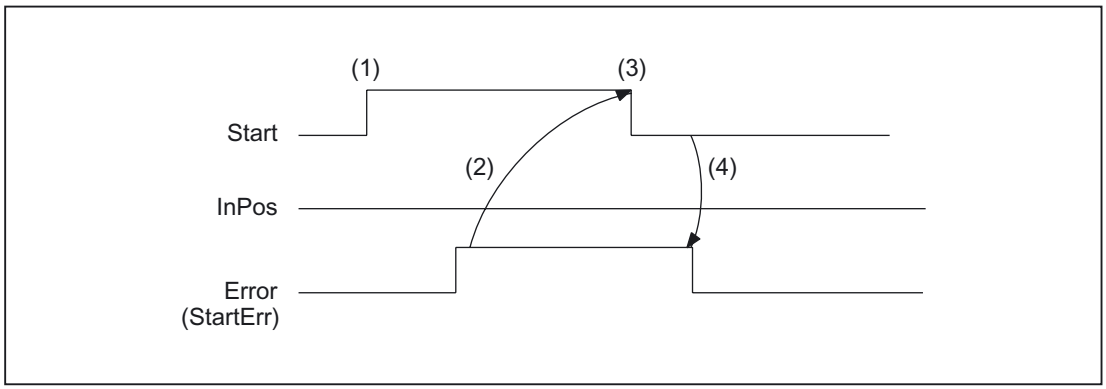
State		Meaning
117	B#16#75	G96/G961 is **not active** in the NC
118	B#16#76	G96/G961 is **still active** in the NC
120	B#16#78	Axis is not an indexing axis (corresponds to NCK alarm 20072)
121	B#16#79	Indexing position error (corresponds to NCK alarm 17510)
125	B#16#7d	DC (shortest distance) not possible (corresponds to NCK alarm 16800)
126	B#16#7e	Absolute value minus not possible (corresponds to NCK alarm 16820)
127	B#16#7f	Absolute value plus not possible (corresponds to NCK alarm 16810)
128	B#16#80	No transverse axis available for diameter programming (corresponds to NCK alarm 16510)
130	B#16#82	Software limit switch plus (corresponds to NCK alarm 20070)
131	B#16#83	Software limit switch minus (corresponds to NCK alarm 20070)
132	B#16#84	Working area limit plus (corresponds to NCK alarm 20071)
133	B#16#85	Working area limit minus (corresponds to NCK alarm 20071)
134	B#16#85	Frame not permitted for indexing axis
135	B#16#87	Indexing axis with "Hirth-toothing" is active (corresponds to NCK alarm 17501)
136	B#16#88	Indexing axis with "Hirth toothing" is active and axis not referenced (corresponds to NCK alarm 17503)
137	B#16#89	Spindle operation not possible for transformed spindle/axis (corresponds to NCK alarm 22290)
138	B#16#8A	The corresponding effective coordinate-system-specific working area limit plus violated for the axis (corresponds to NCK alarm 20082)
139	B#16#8B	The corresponding effective coordinate-system-specific working area limit minus violated for the axis (corresponds to NCK alarm 20082)
System or other serious interrupts:		
200	B#16#c8	Corresponds to system alarm number 450007

Pulse diagram



(1) Activation of function by means of a positive signal edge with start or stop
(2) Positive acknowledgement: Function executed/Position reached
(3) Reset function activation after receipt of acknowledgement
(4) Signal change using FC

Timing diagram (fault scenario)



(1) Activation of function by means of a positive signal edge with start or stop
(2) Negative acknowledgement: Error has occurred
(3) Reset function activation after receipt of acknowledgement
(4) Signal change using FC

Call examples

1. Position spindle:

```
//Positive acknowledgement resets Start:
U M112.0;            //InPos
R M 100.0;           //Start
//Negative acknowledgement, after error evaluation (state: MB114) reset with T12
start
U M113.0;            // Error
U I 6.4;             //Key T12
R M 100.0;           //Start
//Start with T13
U I 6.3;             //Key T13
AN F 112.0;          //Restart only when InPos or Error = 0
AN F 113.0;
S M 100..0;

CALL FC 18 (
     Start :=       M100.0,
     Stop :=        FALSE,
     Funct :=       B#16#1,            //Position spindle
     Mode :=        B#16#2,            //Shortest path
     AxisNo :=      5,
     Pos :=         MD104,
     FRate :=       MD108,
     InPos :=       M112.0,
     Error :=       M113.0,
     State :=       MB114);
```

2. Start spindle rotation:

```
CALL FC 18 (
     Start :=       M100.0,
     Stop :=        FALSE,
     Funct :=       B#16q#,            //Rotate spindle
     Mode :=        B#16#5,            //Direction of rotation as for M4
     AxisNo :=      5,
     Pos :=         0.0,
     FRate :=       MD108,
     InPos :=       M112.0,
     Error :=       M113.0,
     State :=       MB114);
```

3. Start spindle oscillation:

```
CALL FC 18 (
     Start :=       M100.0,
     Stop :=        FALSE,
     Funct :=       B#16#3,            //Oscillate spindle
     Mode :=        B#16#0,
     AxisNo :=      5,
     Pos :=         0.0,
     FRate :=       MD108,
     InPos :=       M112.0,
     Error :=       M113.0,
     State :=       MB114);
```

4. Traverse indexing axis

```
CALL FC 18 (
     Start :=       M100.0,
     Stop :=        FALSE,             //Not used
     Funct :=       B#16#4,            //Traverse indexing axis
     Mode :=        B#16#0,            //Position absolutely
     AxisNo :=      4,
     Pos :=         MD104,             //Default setting in REAL: 1.0;2.0;..
     FRate :=       MD108,
     InPos :=       M112.0,
     Error :=       M113.0,
     State :=       MB114);
```

5. Position axes

```
CALL FC 18 (
     Start :=       M100.0,
     Stop :=        FALSE,             //Not used
     Funct :=       B#16#5,            //Position axes
     Mode :=        B#16#1,            //Position incrementally
     AxisNo :=      6,
     Pos :=         MD104,
     FRate :=       MD108,
     InPos :=       M112.0,
     Error :=       M113.0,
     State :=       MB114);
```

12.15.23 FC 19: MCP_IFM transmission of MCP signals to interface

Function

With the FC MCP_IFM (M-variant) from the machine control panel a range of 19 inches, e.g. MCP 483 are transferred to the corresponding signals of the NCK/PLC interface:

- Modes
- Axis selections
- WCS/MCS switchover commands
- Traversing keys
- Overrides
- Key switch

In the basic program (FC 2) handwheel selections, modes and other operating signals are transferred from the operator panel front (HMI) to the NCK/PLC interface so that the modes support selection from the MCP or HMI.

Transfer of HMI signals to the interface can be deactivated by setting the value of the parameter "MMCToIF" to "FALSE" in FB 1 (DB 7).

The following specifications apply to the **feed override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feed override:**
  - The feed override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feed override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB 5) interface byte if the "Feed override for rapid traverse effective" HMI signal is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this HMI signal.
- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.
  - When the system is switched between MCS and WCS, the active axes are generally deselected.

The **handwheel selection signals from the HMI** are decoded and activated in the machine-axis or the geometry axis interface of the handwheel selected (only if parameter "HWheelMMC := TRUE" in FB 1).

The LEDs on the machine control panel derived from the selections in the acknowledgement.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g. using the appropriate input signals in FC 10: AL_MSG). The associated LEDs are activated at the same time.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

Multiple calls of FC 19 or FC 24, FC 25, FC 26 are permitted in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first call in the cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).

"BAGNo" = 0 or B#16#10 means that the mode group signals are not processed.

"ChanNo" = 0 means that the channel signals are not processed.

The INC selections are transferred to the mode group interface. The activation for this specification is done via the DB10.DBX57.0 (INC-inputs in BAG-area active) through this block once after power up.

Two machine control panels can still be handled in parallel by this module. The module call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st MCP. Support for two MCPs is provided in the control panel blocks up to certain limits (support is not provided as standard for mutual interlocking of axis selections with identical assignments on two MCPs).

Flexible axis configuration

It is possible to be flexible in the assignment of axis selections or direction keys for machine axis numbers.

Better support is now provided by the MCP blocks for the use of two MCPs, which are to run in parallel, in particular for an application using two channels and two mode groups. Note that the axis-numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

To afford this flexibility, tables for axis numbers are stored in DB 10.

For the **first** machine control panel (MCP) the table starts at byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the **second** machine control panel (MCP) starting at byte 32 (symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers must be entered byte-wise here.

It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

For **FC 19**, the **maximum possible number of axis selections** can also be restricted. This upper limit is set for the first Machine control panel in DB10.DBW30 (symbolic name: MCP1MaxAxis) or for the second Machine control panel in DB10.DBW54 (symbolic name: MCP2MaxAxis) set.

The default setting is 0, corresponding to the maximum number of configured axes. The axis numbers and the limit can also be adapted dynamically. Afterwards, a new axis must be selected on FC 19. Axis numbers may not be switched over while the axes are traversing the relevant direction keys.
The compatibility mode is preset with axis numbers **1 to 9** for both MCPs and restricted to the configured number of axes.

Example

More than nine axes are to be controlled with FC19 using a special application. We recommend that you proceed as follows:

- Reserve free key on MCP.
- Evaluate this key as a flipflop.
- Evaluate the flipflop output as pos. and neg. edge.
- For pos. edge write one set of axis numbers in the axis table (DB10) and switch on LED via this key.
- For neg. edge write one set of axis numbers in the axis table (DB10) and switch on LED via this key.

Declaration of the function

```
FUNCTION FC 19: VOID                //symbolic name: MCP_IFM

   VAR_INPUT
       BAGNo :             BYTE ;
       ChanNo:             BYTE ;
       SpindleIFNo:        BYTE ;
   END_VAR

   VAR_OUTPUT
       FeedHold :          BOOL;
       SpindleHold :       BOOL;
   END_VAR

BEGIN
END_FUNCTION
```

Description of formal parameters

The table below shows all formal parameters of the "MCP_IFM" function:

Signal	Type	Type	Range of values	Remark
BAGNo	I	BYTE	0 - b#16#0A and b#16#10 - b#16#1A	No. of mode group to which the mode signals are transferred BAGNo >= b#16#10 means access to the second machine control panel
ChanNo	I	BYTE	0 - B#16#0A	Channel no. for the channel signals
SpindleIFNo	I	BYTE	0 - 31 (B#16#1F)	Number of the axis interface declared as a spindle
FeedHold	O	BOOL		Feed stop from MCP, modal
SpindleHold	O	BOOL		Spindle stop from MCP, modal

MCP selection signals to the user interface

Table 12- 1 Key switch

Source: MCP - Switch	Destination: Interface DB
Position 0	DB10.DBX56.4
Position 1	DB10.DBX56.5
Position 2	DB10.DBX56.6
Position 3	DB10.DBX56.7

Table 12- 2 Operating modes and machine functions

Source: MCP - Key	Destination: Interface DB (parameter BAGNo) Display for BAG 1
AUTOMATIC	DB11.DBX0.0
MDA	DB11.DBX0.1
JOG	DB11.DBX0.2
REPOS	DB11.DBX1.1
REF	DB11.DBX1.2
TEACH IN	DB11.DBX1.0
INC 1 ... 10 000, INC Var.	DB11.DBX2.0 - 2.5

Table 12- 3 Direction keys rapid traverse override

Source: MCP - Key	Destination: Interface DB (parameter ChanNo)
Direction key +	DB21, ...DBX12.7
Direction key -	DB21, ...DBX12.6
Rapid traverse override	DB21, ...DBX12.5
Direction key +	DB21, ...DBX16.7
Direction key -	DB21, ...DBX16.6
Rapid traverse override	DB21, ...DBX16.5
Direction key +	DB21, ...DBX20.7
Direction key -	DB21, ...DBX20.6
Rapid traverse override	DB21, ...DBX20.5

Source: MCP - Key	Destination: Interface DB (all axis DBs)
Direction key +	DB31, ...DBX4.7
Direction key -	DB31, ...DBX4.6
Rapid traverse override	DB31, ...DBX4.5

The transfer is dependent upon the selected axis. The associated interface bits are deleted for axes which are not selected.

Table 12- 4 Override

Source: MCP - Switch	Destination: Interface DB (parameter ChanNo)
Feedrate override	DB21, ...DBB4

Source: MCP - Switch	Destination: Interface DB (all axis DBs)
Feedrate override	DB31, ...DBB0 (selected axis number) The feed override of the 1st MCP is applied to all axes.
Spindle override	DB31, ...DBB19 (parameter SpindleIFNo)

Table 12- 5 Channel signals

Source: MCP - Keys	Destination: Interface DB (parameter ChanNo)
NC start	DB21, ...DBX7.1
NC stop	DB21, ...DBX7.3
RESET	DB21, ...DBX7.7
Single block	DB21, ...DBX0.4

Table 12- 6 Feedrate, spindle signals

Source: MCP - Keys	Destination: FC output parameters
Feed stop Feed enable	Parameter: "FeedHold" linked with memory, LEDs are controlled
Spindle stop Spindle enable	Parameter: "SpindleHold" linked with memory, LEDs are controlled

Checkback signals from user interface for controlling displays

Table 12- 7 Operating modes and machine functions

Destination: MCP - LED	Source: Interface DB (parameter BAGNo) Display for BAG 1
AUTOMATIC	DB11.DBX6.0
MDA	DB11.DBX6.1
JOG	DB11.DBX6.2
REPOS	DB11.DBX7.1
REF	DB11.DBX7.2
TEACH IN	DB11.DBX7.0

Destination: MCP - LED	Source: Interface DB (parameter BAGNo) Display for BAG 1
INC 1 ... 10 000, INC Var.	DB11.DBX8.0 - 8.5

Table 12- 8 Channel signals

Destination: MCP - LED	Source: Interface DB (parameter ChanNo)
NC start	DB21, ...DBX35.0
NC stop	DB21, ...DBX35.2 or DB21, ...DBX35.3
Single block	DB21, ...DBX0.4

Note

Direction key LEDs are controlled by operating the direction keys.

Axis selection and WCS/MCS LEDs are controlled by operating the relevant pushbutton switch.

Call example

```
CALL FC 19(      //Machine control panel M variants Signals to interface
   BAGNo :=           B#16#1,            //Mode group no. 1
   ChanNo :=          B#16#1,            //Channel no. 1
   SpindleIFNo :=     B#16#4,            //Spindle interface number = 4
   FeedHold :=        m22.0,             //Feed stop signal modal
   SpindleHold :=     db2.dbx151.0);     //Spindle stop modal in
                                         //message DB
```

With these parameter settings, the signals are sent to the 1st mode group, the 1st channel and all axes. In addition, the spindle override is transferred in the 4th axis/spindle interface. The feed hold signal is passed to bit memory 22.0 and the spindle stop signal to data block DB2, data bit 151.0.

Reconnecting the axis selections

To ensure a flexible assignment of the axis selection keys to the appropriate axis or spindle, FC 19 **needs not be modified or reprogrammed**. The axis number simply has to be entered in axis table DB10.DBB8 and followed as required: The axis number simply has to be entered in axis table DB10.DBB8 and followed as required:

Example:

The spindle is defined as the 4th axis and must be selected via axis key 9.

```
Solution:
The value 4 must be entered in DB10 byte (8+(9-1)) for the 4th axis.

CALL FC 19(              //Signals to interface
   BAGNo :               B#16#1,       //Mode group no. 1
   ChanNo :              B#16#1,       //Channel no. 1
   SpindleIFNo :         B#16#4,       //Spindle interface number = 4

   FeedHold :            m30.0,        //Feed stop signal modal
   SpindleHold :         m30.1);       //Spindle stop modal
```

12.15.24 FC 21: transfer PLC NCK data exchange

Function

When the Transfer block is called, data are exchanged between the PLC and NCK according to the selected function code. Data are transferred as soon as FC 21 is called, not only at the start of the cycle.

The "Enable" signal activates the block.
FC 21 is processed only when "Enable" = "1".

The following functions for the data exchange between PLC and NCK are supported:

1. Signal synchronized actions at the NCK - channel
2. Signals synchronized actions from NCK - channel
3. Fast data exchange PLC-NCK (Read function in NCK)
4. Fast data exchange PLC-NCK (Write function in NCK)
5. Update control signals to NCK - channel
6. Update control signals to axes (data byte 2 of the user interface)
7. Update control signals to axes (data byte 4 of the user interface)

Declaration of the function

STL representation

```
VAR_INPUT
    Enable : BOOL ;
    Funct: BYTE ;
    S7Var : ANY ;
    IVar1 : INT ;
    IVar2 : INT ;

END_VAR

VAR_OUTPUT
    Error : BOOL ;
    ErrCode : INT ;
END_VAR
```

Explanation of formal parameters

The table below shows all formal parameters of the "Transfer" function.

Signal	Type	Type	Value range	Comment	
Enable	I	BOOL		1 = FC 21 active	
Funct	I	BYTE	1 ... 7	1:	Synchronized actions at channel
				2:	Synchronized actions from channel
				3:	Read data
				4:	Write data
				5:	Control signals to channel
				6, 7:	Control signals to axis
S7Var	I	ANY	S7 data storage area	Depends on "Funct"	
IVAR1	I	INT	0 ...	Depends on "Funct"	
IVAR2	I	INT	1 ...	Depends on "Funct"	
Error	O	BOOL			
ErrCode	O	INT		Depends on "Funct"	

Function 1, 2: Signals synchronized actions to / from Channel

Synchronized actions can be disabled or enabled by the PLC.
The data area is stored on the user interface in DB21, ...DBB300 ...307 (to channel) and DB21, ...DBB308 ...315 (from channel). The parameter "S7Var" is not evaluated for this function, but must be assigned an actual parameter (see call example). The data are transferred to/from the NC as soon as FC 21 is processed.

The following signals are relevant:

Signal	Type	Type	Value range	Comment	
Enable	I	BOOL		1 = FC 21 active	
Funct	I	BYTE	1, 2	1:	Synchronized actions at channel
				2:	Synchronized actions from channel
S7Var	I	ANY	S7 data storage area	Not used	
IVAR1	I	INT	1..MaxChannel	Channel number	
Error	O	BOOL			
ErrCode	O	INT		1:	"Funct" invalid
				10:	Channel no. invalid

Call example:

```
FUNCTION FC 100 : VOID

VAR_TEMP
          myAny: ANY ;
END_VAR

BEGIN
NETWORK

//Deactivate synchronized actions with ID3, ID10 and ID31 in NC channel 1 :
SYAK:     OPEN    DB21;
          SET;
          S       DBX 300.2;      //ID3
          S       DBX 301.1;      //ID10
          S       DBX 303.6;      //ID31
          L       B#16#1;
          T       MB11;
          SPA     TRAN;
```

```
//Synchronized actions from NCK channel 1:
SYVK:     L B#16#2;
          T MB11;
TRAN:     CALL             FC    21 (
               Enable            := M 10.0,          //if TRUE, FC 21 active
               Funct             := MB 11,
               S7Var             := #myAny,          //Not used
               IVAR1             := 1,               //Channel no.
               IVAR2             := 0,
               Error             := M 10.1,
               ErrCode           := MW 12);

END_FUNCTION
```

Function 3, 4: Rapid data exchange PLC-NCK

General

A separate, internal data area is provided to allow the highspeed exchange of data between the PLC and NCK. The size of the internal data field is preset to 4096 bytes. The accesses (read/write) from PLC take place via the FC 21. The occupation of this range (structure) must be defined identically in the NC part program and the PLC user program.

These data can be accessed from the NC parts program by commands $A_DBB[x], $A_DBW[x], $A_DBD[x], $A_DBR[x] (see Parameter Manual System variables).

The concrete address is the data field is specified by a byte offset (0 to 4095) in parameter IVAR1. In this case, the alignment must be selected according to the data format, i.e. a Dword starts at a 4byte limit and a word at a 2byte limit. Bytes can be positioned on any chosen offset within the data field, singlebit access operations are not supported and converted to a byte access operation by FC 21. Data type information and quantity of data are taken from the ANY parameter, transferred via S7Var.

Without additional programming actions, data consistency is only ensured for 1 and 2 byte access in the NCK and in the PLC. For the 2-byte consistency this is true only for the data type WORD or INT, but not for the data type BYTE.

In the case of longer data types or transfer of fields, which should be transferred consistently, a semaphore byte must be programmed in parameter IVAR2 that can be used by FC 21 to determine the validity or consistency of a block. This handling must be supported by the NC, i.e. in the part program, by writing or deleting the semaphore byte. The semaphore byte is stored in the same data field as the actual user data.

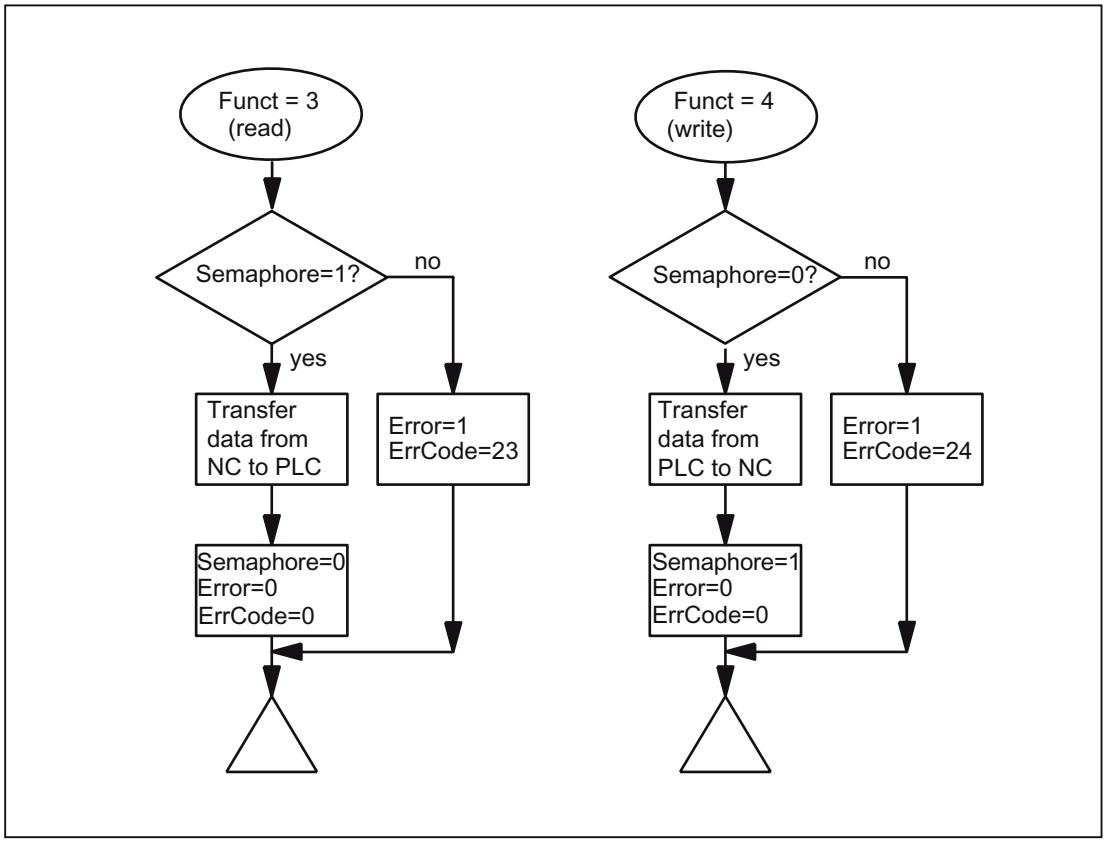
The semaphore byte is identified by a value between 0 and 4095 in IVAR2.

The PLC reads and describes the semaphore byte via FC 21 in the same call, which should transfer the user data. The PLC programmer only needs to set up a semaphore variable. For access from the NC via the parts program, the semaphore feature must be programmed using individual instructions according to the flow chart shown below. The sequence is different for reading and writing variables.

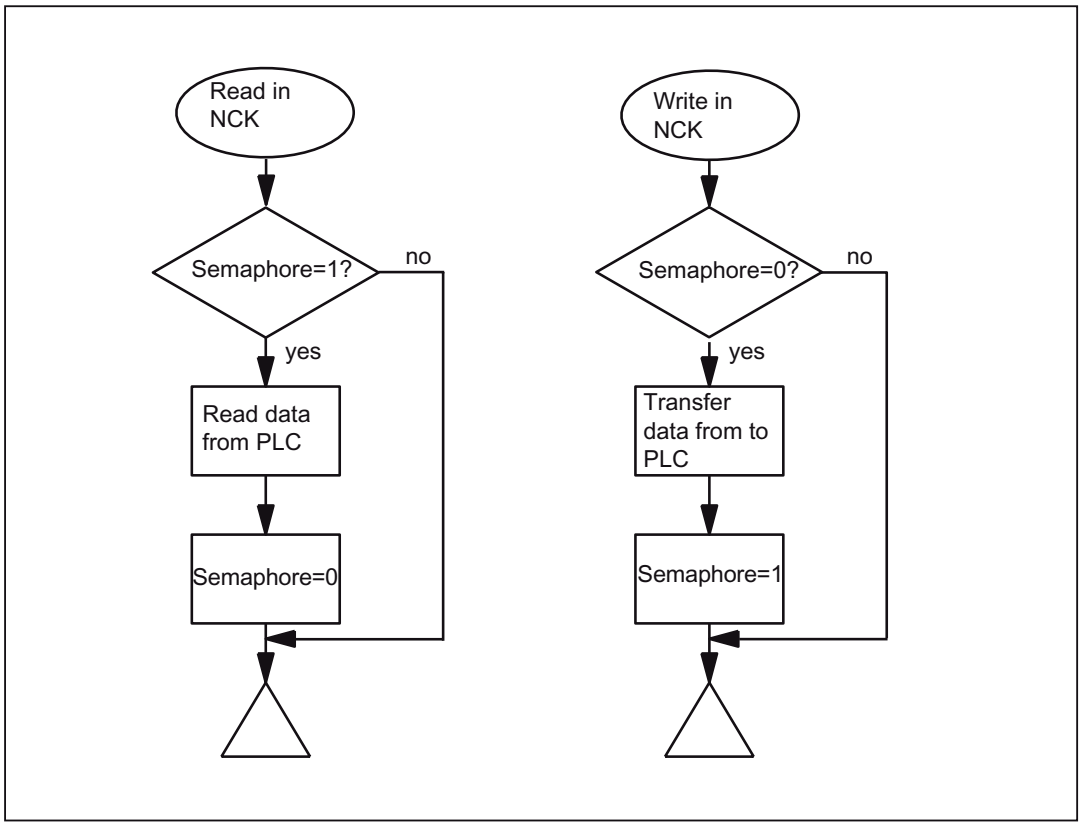
Only individual variables or ARRAYs can be supported directly by the semaphore technique. Structure transfers must be subdivided into individual jobs. The user must ensure data consistency of this structure by programming a semaphore system.

If IVAR2 is set to -1, data are transferred without a semaphore.

Data exchange with semaphore in PLC (schematic of FC21)



Basic structure in NCK:



Variable value ranges

The following signals are relevant:

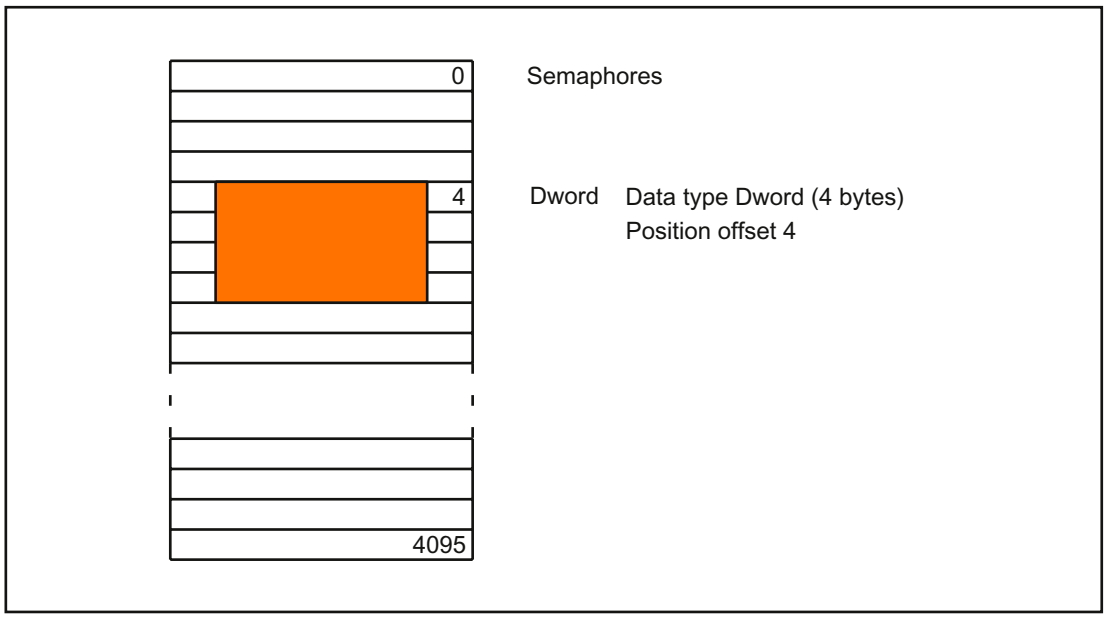
Signal	Type	Type	Value range	Comment	
Enable	I	BOOL		= FC 21 active	
Funct	I	BYTE	3, 4	3:	Read data
				4:	Write data
S7Var	I	ANY	S7 data area, except local data	Source/destination data storage area	
IVAR1	I	INT	0 ... 4095	Position offset	
IVAR2	I	INT	-1 ... 4095	Semaphore byte Transfer without semaphore: -1	
Error	O	BOOL			
ErrCode	O	INT		20:	Alignment error
				21:	illegal position offset
				22:	Illegal semaphore byte
				23:	No new data to be read
				24:	Cannot write data
				25:	Local data parameterized for S7Var

Call example:

1. Read double word of position offset 4 with semaphore in byte 0 and store in MD100:

Data type Dword (4 bytes)

Position offset 4




```
CALL    FC          21 (
          Enable    := M 10.0,                //if TRUE, FC 21 active
          Funct     := B#16#3,                //Read data
          S7Var     := P#M 100.0 DWORD 1,
          IVAR1     := 4,
          IVAR2     := 0,
          Error     := M 10.1,
          ErrCode   := MW12);
UN      M10.1;                  //Enable while 1, until value is read
R       M10.0;
```

Examples: Examples of NCK programming from synchronized actions:

Data transfer from NC to PLC, with data written via synchronized actions;
Byte 0 serves as the semaphore

```
ID=1 WHENEVER $A_DBB[0] == 0 DO $A_DBR[4] = $AA_IM[X] $A_DBB[0] = 1
```

Data transfer from PLC to NC, with data read via synchronized actions;
Byte 1 serves as the Semaphore:

```
ID=2 WHENEVER $A_DBB[1] == 1 DO $R1 = $A_DBR[12] $A_DBB[1] = 0
```

2. Read word of position offset 8 without semaphore and store in MW 104:

```
CALL   FC           21 (

         Enable     :=M 10.0,                  //if TRUE, FC 21 active
         Funct      :=B#16#3,                  //Read data
         S7Var      :=P#M 104.0 WORD 1,
         IVAR1      :=8,
         IVAR2      :=-1,
         Error      :=M 10.1,
         ErrCode    :=MW12);
```

Function 5: Update control signals to channel

The purpose of this function is to transmit important control signals at high speed in between cyclic data transfers. Data bytes 6 and 7 of user interfaces DB21, ... are transferred to the NC. The channel is specified in parameter "IVAR1". This enable, for example, the feed disable, read-in disable to be transferred outside of the PLC cycle.

The following signals are relevant:

Signal	Type	Type	Value range	Comment	
Enable	I	BOOL		1= FC 21 active	
Funct	I	BYTE	5	5: Control signals to channel	
S7Var	I	ANY	S7 data storage area	Not used	
IVAR1	I	INT	1. Max. channel	Channel number	
Error	O	BOOL			
ErrCode	O	INT		1:	"Funct" invalid
				10:	Channel no. invalid

Function 6: Update control signals to axes

The purpose of function 6 is to transmit important control signals at high speed in between cyclic data transfers. The **data byte 2** of application interface DB31, ... is transferred to the NC. The transfer is performed for all activated axes. This allows the controller enable to be transferred outside the PLC cycle, for example.

The following signals are relevant:

Signal	Type	Type	Value range	Comment
Enable	I	BOOL		1= FC 21 active
Funct	I	BYTE	6	6: Control signals to axes
S7Var	I	ANY	S7 data storage area	Not used
IVAR1	I	INT	0	
Error	O	BOOL		
ErrCode	O	INT		1: "Funct" invalid

Function 7: Update control signals to axes

The purpose of function 7 is to transmit important control signals at high speed in between cyclic data transfers. The **data byte 4** of application interface DB31, ... is transferred to the NC. The transfer is performed for all activated axes. This enables, for example, the feed stop to be transferred outside the PLC cycle.

The following signals are relevant:

Signal	Type	Type	Value range	Comment
Enable	I	BOOL		1= FC 21 active
Funct	I	BYTE	7	7: Control signals to axes
S7Var	I	ANY	S7 data storage area	Not used
IVAR1	I	INT	0	
Error	O	BOOL		
ErrCode	O	INT		1: "Funct" invalid

12.15.25 FC 22: TM_DIR Direction selection for tool management

Function

The block TM_DIR provides the shortest path for positioning a magazine or a revolver based on the actual and setpoint position.

As long as a 1 signal is applied to the **Start** input, all output parameters are updated cyclically. Changes can be made to input parameters (e.g. position values) in subsequent PLC cycles.
The output signals are undefined when the start signal is at 0 level.

In the case of direction selection with special positioning input "Offset" > 0, a new setpoint position is calculated from the setpoint and special positions and the number of magazine locations according to the following formula:

New setpoint position = (setpoint pos. - (special pos. -1)) neg. modulo # locations

The new setpoint position corresponds to the location number at which the magazine must be positioned so that the setpoint position requested by the user corresponds to the location number of the special position. The directional optimization is active both with and without special positioning.

The block must be called once with the appropriate parameter settings for each magazine.

⚠ WARNING

The block may only be called in conjunction with the tool management.

Note

For further details on tool management (also with regard to PLC) refer to the Description of Functions Tool Management. Furthermore, PI services are provided for tool management via the FB 4, FC 7 and FC 8 (see also the corresponding Sections in this documentation).

Declaration of the function

STL representation

```
FUNCTION FC 22 : VOID
// NAME:             TM_DIR
VAR_INPUT
    MagNo:           INT ;
    ReqPos:          INT;
    ActPos:          INT;
    Offset:          BYTE ;
    Start:           BOOL ;
END_VAR

VAR_OUTPUT
    Cw:              BOOL ;
    Ccw:             BOOL ;
    InPos:           BOOL ;
    Diff:            INT;
    Error :          BOOL ;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

The table below shows all formal parameters of the "TM_DIR" function.

Signal	Type	Type	Range of values	Remark
MagNo	I	INT	1 ...	Magazine number
ReqPos	I	INT	1 ...	Setpoint location
ActPos	I	INT	1 ...	Actual location
Offset	I	BYTE	0 ...	Offset for special positioning
Start	I	BOOL		Start of calculation
Cw	A	BOOL		1 = Move magazine clockwise
Ccw	A	BOOL		1 = Move magazine counterclockwise
InPos	A	BOOL		1 = In position
Diff	A	INT	0 ...	Differential path (shortest path)
Error	A	BOOL		1 = error

Call example

```
CALL FC 22 (                    //Tool management direction selection
   MagNo :=        2,           //Magazine number
   ReqPos :=       mw 20,       //Target position
   ActPos :=       mw 22,       //Current position
   Offset :=       b#16#0,      //Offset for special positioning
   Start :=        m 30.4,      //Start trigger
                                //Return parameters
   Cw :=           m 30.0,      //Move magazine
                                //in anticlockwise direction
   Ccw :=          m 30.1,      //Move magazine
                                //in anticlockwise direction
   InPos :=        m 30.2,      //Magazine in position
   Diff :=         mw 32,       //Differential path
   Error :=        m30.3        //Error has occurred
   );
```

12.15.26 FC 24: MCP_IFM2 transmission of MCP signals to interface

Function

With FC MCP_IFM2 (M variant slimline machine control panel, e.g. MCP 310), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Modes
- Axis selections
- WCS/MCS switchover
- Traversing keys
- Overrides or override simulation signals

In the basic program (FC 2), handwheel selections, modes and other operating signals continue to be transferred from the operator panel front or HMI to the NCK/PLC interface so that the modes support selection from the MCP or HMI.

Transfer of HMI signals to the interface can be deactivated by setting the value of the parameter "MMCToIF" to "FALSE" in FB 1 (DB 7). "MMCToIF" can also be activated/deactivated in the cyclic program by setting and resetting (e.g. R gp_par.MMCToIF).

The following specifications apply to the **feed override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feed override:**
  - The feed override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feed override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB 5) interface byte if the "Feed override for rapid traverse effective" HMI signal is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this HMI signal.
- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.
  - When the system is switched between MCS and WCS, the active axes are generally deselected.

The **handwheel selection signals from HMI** are decoded and activated in the machine axis or the geometry axis interface of the handwheel selected (only if parameter "HWheelMMC: = TRUE" in FB 1).

The associated LEDs on the machine control panel are derived from the acknowledgements from the relevant selections.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g. using the appropriate input signals in FC 10: AL_MSG). The associated LEDs are activated at the same time.

The **spindle direction** (+, -) is not switched directly either, but made available as output parameter "SpindleDir" permitting, for example, FC 18 to be parameterized. A spindle enable signal is also switched via parameter "SpindleHold". One possible method of moving a spindle directly is to preselect it as an axis so that it can be traversed via (axis) direction keys.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

Multiple calls of FC 24 or FC 19, FC 25, FC 26 are permitted in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first call in the cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).

"BAGNo" = 0 or B#16#10 means that the mode group signals are not processed.

ChanNo = 0 means that the channel signals are not processed.

The INC selections are transferred to the mode group interface. The activation for this specification is done via the DB10.DBX57.0 (INC-inputs in BAG-area active) through this block once after power up.

Furthermore, two machine control panels can be handled in parallel by this block. The module call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st MCP Support for two MCPs is provided in the control panel blocks up to certain limits (support is not provided as standard for mutual interlocking of axis selections with identical assignments on two MCPs).

Flexible axis configuration

It is possible to be flexible in the assignment of axis selections or direction keys for machine axis numbers.

Better support is now provided by the MCP blocks for the use of two MCPs, which are to run in parallel, in particular for an application using two channels and two mode groups. Note that the axis-numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

To afford this flexibility, tables for axis numbers are stored in DB 10.
For the **first** machine control panel (MCP), the table starts at byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the **second** machine control panel (MCP) starting at byte 32 (symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers must be entered byte by byte here. It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

For **FC 24**, the **maximum possible number of axis selections** can also be restricted.
This upper limit is set for the first machine control panel in DB10.DBW30 (symbolic name: MCP1MaxAxis) or for the second machine control panel in DB10.DBW54 (symbolic name: MCP2MaxAxis) for the respective MCP.

The default setting is 0, corresponding to the maximum number of configured axes. The axis numbers and the limit can also be adapted dynamically. Afterwards, a new axis must be selected on FC 24. Axis numbers may not be switched over while the axes are traversing the relevant direction keys. The compatibility mode is preset with axis numbers **1 to 6** for both MCPs and restricted to the configured number of axes.

Declaration of the function

```
FUNCTION FC 24: VOID
// NAME:                  MCP_IFM2

VAR_INPUT
    BAGNo :               BYTE ;
    ChanNo:               BYTE ;
    SpindleIFNo:          BYTE ;
END_VAR

VAR_OUTPUT
    FeedHold :            BOOL;
    SpindleHold :         BOOL;
    SpindleDir:           BOOL;
END_VAR

BEGIN
END_FUNCTION
```

Description of formal parameters

The table below shows all formal parameters of the "MCP_IFM2" function:

Signal	Type	Type	Range of values	Remark
BAGNo	I	BYTE	0 - b#16#0A and b#16#10 - b#16#1A	No. of mode group to which the mode signals are transferred BAGNo >= b#16#10 means access to the second machine control panel
ChanNo	I	BYTE	0 - B#16#0A	Channel no. for the channel signals
SpindleIFNo	I	BYTE	0 - 31 (B#16#1F)	Number of the axis interface declared as a spindle
FeedHold	O	BOOL		Feed stop from MCP, modal
SpindleHold	O	BOOL		Spindle stop from MCP, modal
SpindleDir	O	BOOL		Direction of spindle rotation
				0: Corresponds to + (left)
				1: Corresponds to - (right)

Call example

```
CALL FC 24(                                 //Slimline machine control panel M variants
                                            //Signals to interface
     BAGNo :=           B#16#1,             //Mode group no. 1
     ChanNo :=          B#16#1,             //Channel no. 1
     SpindleIFNo :=     B#16#4,             //Spindle interface number = 4
     FeedHold :=        m22.0,              //Feed stop signal modal
     SpindleHold :=     db2.dbx151.0,       //Spindle stop modal in message data block
     SpindleDir:=       m22.1);             //Spindle direction return
```

With these parameter settings, the signals are sent to the 1st mode group, the 1st channel and all axes. In addition, the spindle override is transferred in the 4th axis/spindle interface. The feed hold signal is passed to bit memory 22.0 and the spindle stop signal to data block DB2, data bit 151.0. The spindle direction feedback signal supplied via parameter "SpindleDir" can be used as a direction input for an additional FC 18 call.

12.15.27 FC 25: MCP_IFT transfer of MCP/OP signals to interface

Function

With the FC MCP_IFM (M variant) from the machine control panel a range of 19 inches, e.g. MCP 483 are transferred to the corresponding signals of the NCK/PLC interface:

- Modes
- Direction keys of four axes
- WCS/MCS switchover commands
- Overrides
- Key switch

In the basic program (FC 2), handwheel selections, modes and other operating signals continue to be transferred from the operator panel front or HMI to the NCK/PLC interface so that the modes support selection from the MCP or BT.

Transfer of HMI signals to the interface can be deactivated by setting the value of the parameter "MMCToIF" to "FALSE" in FB 1 (DB 7).

The following specifications apply to the **feed override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feed override:**
  - The feed override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feed override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB 5) interface byte if the "Feed override for rapid traverse effective" HMI signal is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this HMI signal.
- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.

The **handwheel selection signals from the HMI** are decoded and activated in the machine axis or the geometry axis interface of the handwheel selected (only if parameter "HWheelMMC := TRUE" in FB 1). = TRUE").

The associated LEDs on the machine control panel derived from the acknowledgements of the relevant selections.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g. using the appropriate input signals in FC 10: AL_MSG). The associated LEDs are activated at the same time.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

Multiple calls of FC 25 or FC 19, FC 24, FC 26 are permitted in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).

"BAGNo" = 0 or B#16#10 means that the mode group signals are not processed.

ChanNo = 0 means that the channel signals are not processed.

Flexible axis configuration

It is possible to be flexible in the assignment of axis selections or direction keys for machine axis numbers.

Support is now provided by the MCP blocks for the use of two MCPs, which are operated simultaneously, in particular for an application using two channels and two mode groups. The module call for the second machine control panel in OB1 cycle must come after the call of the first MCP Note that the axis numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

To achieve this flexibility, tables for axis numbers are stored in DB 10.
For the **first** machine control panel (MCP), the table starts at byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the **second** machine control panel (MCP) starting at byte 32 (symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers are entered here bytewise. It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

The restriction of the **possible number of axes at FC 25** is done via the 0-values in the axis table. The axis numbers can also be adapted dynamically. During the manual traversing of axes using the direction keys, the axis numbers must not be switched over. The compatibility mode is preset with axis numbers **1 to 4** for both MCPs and restricted to the configured number of axes.

Note

For further information see "FC 19: MCP_IFM transmission of MCP signals to interface (Page 1131) ".

Declaration of the function

```
FUNCTION FC 25: VOID
// NAME:                MCP_IFT

VAR_INPUT
    BAGNo :              BYTE ;
    ChanNo:              BYTE ;
    SpindleIFNo:         BYTE ;
END_VAR

VAR_OUTPUT
    FeedHold :           BOOL;
    SpindleHold :        BOOL;
END_VAR

BEGIN
END_FUNCTION
```

Description of formal parameters

The table below shows all formal parameters of the "MCP_IFT" function:

Signal	Type	Type	Range of values	Remark
BAGNo	I	BYTE	B#16#00 - B#16#0A	**1st MCP**: Mode group interface in which the mode signals are transferred
			B#16#10 - B#16#1A	**2nd MCP**: Mode group interface in which the mode signals are transferred
ChanNo	I	BYTE	B#16#00 - B#16#0A	Channel no. for the channel signals
SpindleIFNo	I	BYTE	B#16#00 - B#16#1F	Axis interface in which the spindle data is transferred
FeedHold	O	BOOL		Feed stop from MCP, modal
SpindleHold	O	BOOL		Spindle stop from MCP, modal

Call example

```
CALL FC 25(                                 //Machine control panel T variants
                                            //Signals to interface
   BAGNo :=              B#16#1,            //Mode group no. 1
   ChanNo :=             B#16#1,            //Channel no. 1
   SpindleIFNo :=        B#16#4,            //Spindle interface number = 4
   FeedHold :=           m22.0,             //Feed stop signal modal
   SpindleHold :=        db2.dbx151.0);     //Spindle stop modal in message data block
```

With these parameter settings, the signals are sent to the 1st mode group, the 1st channel and all axes. In addition, the spindle override is transferred in the 4th axis/spindle interface. The feed hold signal is passed to bit memory 22.0 and the spindle stop signal to data block DB2, data bit 151.0.

12.15.28 FC 26: HPU_MCP transmission of HT8 signals to interface

Function declaration

```
FUNCTION FC 26: VOID
// NAME:           HPU_MCP

VAR_INPUT
    BAGNo :        BYTE ;
    ChanNo:        BYTE ;
END_VAR

BEGIN
END_FUNCTION
```

Parameter

Parameter	Type	Type	Range of values	Remark
BAGNo	I	BYTE	1. MCP: B#16#00 - B#16#0A 2. MCP: B#16#10 - B#16#1A	Upper nibble: Number of the MCP whose signals are to be transferred. 0 = 1. MCP, 1 = 2. MCP Lower nibble: Number of the mode group, in which the mode group-specific interface signals are to be transferred. The mode group-specific signals are not processed, if the mode-group number is 0.
ChanNo	I	BYTE	B#16#00 - B#16#0A	Number of the channel, in which the channel-specific interface signals are to be transferred. The channel-specific signals are not processed, if the channel number is 0.
Type: I = input parameter, O = output parameter				

Call examples

Call of the FC 26 for the first MCP, the first mode group and the first channel of the NC.

```
CALL FC 26(                             //Machine control panel of HT8
             BAGNo :=      B#16#01,     //1.MCP, 1.BAG
             ChanNo :=     B#16#01);    //Channel 1
```

Call of the FC 26 for the second MCP, the second mode group and the third channel of the NC.

```
CALL FC 26(                             //Machine control panel of HT8
             BAGNo :=      B#16#12,     //2.MCP, 2.BAG
             ChanNo :=     B#16#03 );   //Channel 3
```

General function description

The function FC 26 "HPU_MCP (nachine control panel-signals of the hand-held unit HT8" transfers the HT8-specific signals of the following functions between the HT8-input/output data areas parameterized in the function block FB 1 (Parameter: MCPxIn and MCPxOut) and the NC/PLC-interface:

- Modes
- Machine function INC
- Coordinate system WCS or MCS
- Axial traverse key
- Axis selection
- Feed override
- Rapid traverse override
- Keyswitch information

Note

Mode switchover through HT 8 and/or HMI

The function FC 2 "GP_HP Basic program, cyclic part" transfers the signals of the block-switchover in such a way that an alternative selection of MCP of HT 8 and of the HMI is possible. The transfer of the HMI signals to the NC/PLC interface can also be switched off in the function block FB 1 with the parameter "MMCToIF" = FALSE .

Active axes:

Using HT 8 a maximum of 6 axes can be addressed at the same time. The selection of the axes is to be realized by the user/machine manufacturer in the PLC user program.

Flexible axis configuration

The function FC 26 enables a flexible assignment of the machine axes to the traversing keys or to the axis selection. 2 tables are available in DB 10 for this purpose:

- Machine axis table 1. MCP: DB10.DBB8 to DBB13 (Table of the machine axis number) Symbolic name: MCP1AxisTbl[1..22]
- Machine axis table 2. MCP: DB10.DBB32 to DBB37 (Table of the machine axis number) Symbolic name: MCP2AxisTbl[1..22]

In the tables the axis numbers n (with n = 1, 2, ...) of the active machine axis are to be entered byte-wise. The value 0 must be entered in the unused table locations.

The table length can be specified to the FC 26:

- 1st MCP: DB10.DBB30 (upper limit of the machine axis table)
- 2nd MCP: DB10.DBB54 (upper limit of the machine axis table)

A value of 4, for example, means that FC 26 takes into account only the first 4 table entries or machine axes. The maximum value for the FC 26 is 6. For value 0 or values greater than 6 the maximum value is taken implicitly.

Note

Please note the following constraints:

- A check of the permissible machine axis numbers is not done. Invalid machine axis numbers can lead to a PLC stop.
- The machine axis numbers can be changed dynamically. The table may not be written, if currently a machine axis is being moved via a traversing key.

Transfer of the traversing key signals depending upon the active coordinate system

The traversing key signals for 6 axes lie in the HT 8 input data area below:

- EB n + 2, Bit 0 - Bit 5 (positive traversing direction)
- EB n + 3, Bit 0 - Bit 5 (negative traversing direction)

The switchover of the coordinate system is done via the input signal:

- EB n + 0, Bit 0 (MCS/WCS)

The input signal is evaluated in FC 26 with the help of the edge trigger flag. The active coordinate system is shown in the following output signal:

- AB n + 0, Bit 0 (MCS/WCKS) with 0 = MCS, 1 = WCS

In case of active MCS the traversing key signals of the axes 1 - 6 are transferred in the axis-specific interfaces (DB31, ...DBX4.6 and DBX4.7 (traversing key +/-)) of the axes specified in the machine axis tables (DB10.DBB8 to DBB13 or DBB32 to DBB37).

In case of active WCS it is assumed that the axes 1 - 3 of the machine axis table are geometric axes. For this reason the traversing key signals:

- Of the axes **1** - **3** (EB n + 2 / 3, Bit 0 - Bit 2) are transferred in the interface of the geometric axes in DB21, ...DBB 12 + (n * 4), with n = 0, 1, 2), Bit 6 and Bit 7 (traversing keys +/-) of the channel specified with the parameter "ChanNo" .
  The assignment of the traversing key signals of the axes 1, 2 and 3 to the geometric axes 1, 2 and 3 of the channel is permanent and may not be changed.
- Of the axes **4** - **6** (EB n + 2 / 3, Bit 3 - Bit 5) are transferred in the axis-specific interface (DB31, ...DBX4.6 and DBX4.7 (traversing keys +/-)) of the axes 4 - 6 entered in the machine axis table (DB10.DBB**11** to DBB**13** or DBB**35** to DBB**37**).

No traversing of machine axes in WCS

In case of active WCS (AB n + 0, Bit 0 = 1) the traversing of the machine axes can be locked. For this, the following output signals are to be set in the PLC user program:

- AB n + 3, Bit 7 = 1 (For WCS: no machine axes)

  Requirement to the FC 26, not to transfer any traversing key signals for the machine axes. The traversing key signals for the axes 1 - 3 of the machine axis table are transferred to the geometric axes 1 - 3 of the specified channel. The traversing key signals for the axes 4 - 6 of the machine axis table are not transferred.

- AB n + 2, Bit 6 (axes 7 - n selected)

  Requirement at the FC 26 not to transfer any traversing key signals, since the axes 1 - 6 of the machine axis table are switched over. The axes 1 - 3 are thus not geometric axes, but instead also machine axes.

Feed override

The value of the HT8 override switch is transferred as feed override in the chanel-specific interface DB 21, ...DBB4 (feedrate override) of the programmed channel (parameter: "ChanNo") and in the axis-specific interfaces DB31, ...DBB0 (feedrate override) of the axes programmed in the table DB10.DBB8 to DBB13 (machine axis number).

Rapid traverse override

Is for the programmed channel (parameter: "ChanNo") the signal DB21, ...DBX25.3 = 1 (feedrate override for rapid traverse) set, the value of the HT8 override switch is set as rapid traverse override in this channel-specific. Interface in DB 21, ...DBB5 (rapid traverse override) and in addition the signal DB21, ...DBX6.6 = 1 (rapid traverse override active) is set.

Machine function INC

The HT8 signals of the machine functions INC are transferred differently depending upon the active coordinate system MCS or WCS:

- Active coordinate system: MCS

  The selected machine function INC is transferred for all 6 axes in the axis-specific interfaces in DB31, ...DBX5.0 to DBX5.5 (machine function) of the axes programmed in the table in DB10.DBB8 to DBB13 (machine axis numbers) .

- Active coordinate system: WCS

  For the axes 1 to 3 the signals of the machine function INC are transferred in the channel-specific. Interface in DB21, ...DBX13.0 to DBX13.5 (machine function) of the programmed channel (Parameter: "ChanNo").

  For the axes 4 to 6 the signals of the machine function INC are transferred in the channel-specific. interfaces in DB31, ...DBX5.0 to DBX5.5 (machine function) of the axes programmed in the table in DB10.DBB11 to DBB13 (machine axis numbers).

The selection signals of the INC machine functions are transferred in the mode group-specific interface DB11 DBB 2 + (n * 20), Bit 0 to Bit 5 (with n = 0, 1, 2, ...). The FC 26 informs the NCK about the activation of the mode group-interface for the INC machine function once after the power-up with DB10.DBX57.0 (INC inputs active in the mode group area).

Handwheel selection

The hand-wheel selection signals are evaluated by HMI and transferred to the corresponding NC/PLC interface signals of the machine or geometric axes:

- Geometry axes: DB21, ... DBB 12 + (n * 4), Bit 0 to Bit 2 (with n = 0, 1, 2)
- Machine axes: DB31, ...DBX4.0 to DBX4.2

Requirement: FB 1 parameter: "HWheelMMC" = TRUE

Multiple call in one PLC cycle

Multiple calls of FC 26 are permitted in a single PLC cycle. Upon the first call in the PLC cycle:

- All actions of the parameterized blocks are executed
- The LED signals are written in the output area
- In case of selected WCS, the traversing key signals of the geometric axes are written
- The signals for the selection and deselection of the individual block are processed

Upon further calls of the FC 26 only a reduced processing of the channel and mode group-interface is done.

Processing of two MCP

If the function FC 26 is called twice for two MCP in the cyclic sequence of the PLC program (organization block OB 1), the call for the second MCP must be made after the call for the first MCP.

Note

If an axis can be traversed from two MCP, then the implementation of a mutual interlocking is the responsibility of the user (machine manufacturer).

Failure of the MCP of HT8

In case of failure of the MCP of HT8 all the input signals are set to the value 0.

12.15.28.1 Overview of the NC/PLC interface signals of HT 8

Operating modes and machine functions

Source: MCP	Destination: Programmed mode group (Parameter BAGNo) Display for BAG 1
AUTOMATIC	DB11.DBX0.0
MDI	DB11.DBX0.1
JOG	DB11.DBX0.2
REPOS	DB11.DBX1.1
REF	DB11.DBX1.2
TEACH IN	DB11.DBX1.0
INC 1 ... 10 000, INC Var.	DB11.DBX2.0 - DBX 2.5

Traversing keys and rapid traverse override

Source: MCP	Aim: Geometry axis of the prog. channel (Parameter: ChanNo)
Traversing key +	DB21, ...DBX12.7
Traversing key -	DB21, ...DBX12.6
Rapid traverse override	DB21, ...DBX12.5
Traversing key +	DB21, ...DBX16.7
Traversing key -	DB21, ...DBX16.6
Rapid traverse override	DB21, ...DBX16.5
Traversing key +	DB21, ...DBX20.7
Traversing key -	DB21, ...DBX20.6
Rapid traverse override	DB21, ...DBX20.5

Source: MCP	Aim: Prog. axes corresponding to the table in DB 10, DBB 8 - 13 (1. MCP) or DBB 32 - 37 (2. MCP)
Traversing key +	DB31, ...DBX4.7
Traversing key -	DB31, ...DBX4.6
Rapid traverse override	DB31, ...DBX4.5

Override

Source: MCP	Aim: Programmed channel (Parameter: ChanNo)
Feed override	DB21, ...DBB4

Source: MCP	Aim: Prog. axes corresponding to the table in DB 10, DBB 8 - 13 (1. MCP) or DBB 32 - 37 (2. MCP)
Feed override	DB31, ...DBB0

Channel signals

Source: MCP	Aim: Programmed channel (Parameter: ChanNo)
NC start	DB21, ...DBX7.1
NC stop	DB21, ...DBX7.3
RESET	DB21, ...DBX7.7
Single BLock	DB21, ...DBX0.4

12.15.28.2 Overview of the NC/PLC interface signals of HT 8

Operating modes and machine functions

Destination: MCP	Source: Interface-DB (Parameter BAGNo) Display for BAG 1
AUTOMATIC	DB11.DBX6.0
MDA	DB11.DBX6.1
JOG	DB11.DBX6.2
REPOS	DB11.DBX7.1
REF	DB11.DBX7.2
TEACH IN	DB11.DBX7.0

12.15.29 FC 19, FC 24, FC 25, FC 26 source code description

Task

Machine control panel to application interface (FC 19 M variant, FC 24 slim variant, FC 25 T variant, FC 26 HT8 variant)

Associated blocks

DB 7, no. of MOGs, channels, axes

DB 7, pointer of machine control panel

DB 8, storage for the next cycle

Resources used

None.

General

The blocks FC 19 (M version), FC 24 (slim-line version), FC 25 (T version) and FC 26 (HT8 version) transfer the signals of the machine control panel to and from the application interface. In the input parameters, "ModeGroupNo" selects the mode group to be processed by the block. The "ModeGroupNo" parameter also selects the number of the machine control panel (Bit 4). "ChanNo" selects the channel to be processed.

Not FC 26:

The "SpindleIFNo" parameter defines the axis interface of the spindle. The spindle override is transferred to this spindle interface. The parameters are checked for incorrect parameterization.

Not FC 26:

Output parameters "FeedHold" and "SpindleHold" are generated from the 4 feed/spindle disable and feed/spindle enable keys and are returned with "logical 1" for disable.

Information for the next cycle is stored in DB8, bytes 0 to 3 or bytes 62 to 65, depending on the machine control panel number. This information is the edge trigger flag, feed value and selected axis number. The blocks are provided with user data via the pointer parameters in DB 7 "MCP1In" and "MCP1Out" ("MCP2In" and "MCP2Out"). The pointers are addressed indirectly via a further pointer from the VAR section of DB7 in order to avoid absolute addressing. This additional pointer is determined symbolically in FB1.

Block Description

All 4 components have a similar structure and are classified for the individual subtasks:

In the Input network, various parameters are copied to local variables. The machine control signals (user data for input/output area) are also copied between locations using the various pointers in DB 7 (gp_par). These local variables are handled in the block for reasons of efficiency. Some values are initialized for the startup.

MCS/WCS switchover with edge evaluation, axis selections, direction keys and rapid traverse overlay is determined in the **Global_IN network** for further processing in the block User-specific changes must take place in this part of the program, which are mainly oriented at the axis selection.

Only the keyswitch information is copied in **Network NC** .

The **mode group network** transfers the modes of the keys as dynamic signals to the NCK. The INC checkback signals from the NC are stored temporarily for the corresponding LEDs. If the mode group number is 0, this network is not processed. A too large number generates the message 401901 or 402501 and changes over after stop.

In the **Channel network** the NC Start, Stop, Reset and Single Block functions are activated by corresponding checkback signals. The direction keys of the geometry axes are supplied if a corresponding preselection is made, otherwise they are cleared. If the channel number is 0, this network is not processed. A too large number generates the message 401902 or 402502 and changes over after stop.

The **Spindle network** transfers the spindle override to the interface configured via "SpindleIFNo".

The **Network Axes** transfers the feed override to the selected axis interface. The direction keys are assigned to the selected axis/spindle. If an axis has been selected previously, the direction information is set to 0.

The output parameters are prepared and the LED signals of the INC machine function are generated in the **Global_OUT network** .

The **Output network** transfers the output signals of the machine control panel from the VAR_TEMP image to the logical address. The data for the next cycle are also saved.

Axis selection extension

The Global_IN network must be modified if more than nine axes are selected. If other keys and LEDs are to be used on the machine control panel here, proceed as follows:

1. The command UD DW#16#Value (comment: Clear all axis LEDs for display) deletes all defined LEDs for axis selections. The bit mask is currently processing the nine axis selection LEDs.
2. The command UW W#16# (comment: ”Masking all the axis selection buttons”) checks whether the direction has changed. The bit string must be adjusted here.
3. The branch destination list (SPL) must be expanded with new jump labels. The new jump labels should be inserted in descending order before label m009. The selection information should be extended for the new jump labels, as described for labels m009 and m008.

Note

The blocks are made available as STL sources if required. But they do not match the current status of the block. Some details of the actual implementation in C have been developed further. For this reason we recommend that you specify your additional requirements for the blocks and that you pass these onto project management via the sales department.

12.15.30 FC 1005: AG_SEND transfers data to Ethernet CP

Function

The FC block AG_SEND transfers data to the Ethernet CP for transfer via a configured connection.

The specified functions correspond to the functions of the library "SIMATIC_NET_CP" of the S7-300 CPU in STEP 7. In general, the online help of these functions applies for these functions and therefore a detailed description is not provided here.

The functions AG_SEND, AG_RECV can be used for data exchange with another station via the integrated "CP 840D sl".

Description of formal parameters

The following table shows the formal parameters of the function AG_SEND.

Signal	Type	Type	Value range	Remark
ACT	I	BOOL		Job initiated
ID	I	INT		Connection ID
LADDR	I	WORD		Module start address (special SINUMERIK feature; see description below the table)
SEND	I	ANY		Specifies the address and length. The address of the data area alternatively refers to: • Bit memory address area • Data block area
LEN	I	INT		Number of bytes, which should be sent with the job from the data area
DONE	O	BOOL		Job successfully completed
ERROR	O	BOOL		Error display
STATUS	O	WORD		Status display

When using the functions AG_SEND and AG_RECV, data is transported to the communication partner via the Ethernet bus of the CP. The communication partner is configured in STEP 7 in the "NetPro" tool.

The special feature when calling the functions involves the specification of parameter "LADDR" at the named blocks. In the case of SINUMERIK 840D sl, value W#16#8110 must be connected to parameter "LADDR".

In the basic program, this function is available under the FC number 1005 (this FC corresponds to the FC number FC 5 in the library "SIMATIC_NET_CP").

The block can also be used in a SIMATIC-CPU 3xx with CP343-1.

The protocols TCP and UDP are supported. TCP is the preferred protocol.

Note

With the function AG_SEND, parameter ACT must be TRUE until a result is signaled in DONE or ERROR.

12.15.31 FC 1006: AG_RECV receives data from the Ethernet CP

Function

The FC block AG_RECV receives data transferred via a configured connection from the Ethernet CP.

The specified functions correspond to the functions of the library "SIMATIC_NET_CP" of the S7-300 CPU in STEP 7. In general, the online help of these functions applies for these functions and therefore a detailed description is not provided here.

The functions AG_SEND, AG_RECV can be used for data exchange with another station via the integrated "CP 840D sl".

Description of formal parameters

The following table shows the formal parameters of the function AG_RECV.

Signal	Type	Type	Value range	Remark
ID	I	INT		Connection ID
LADDR	I	WORD		Module start address (special SINUMERIK feature; see description below the table)
RECV	I	ANY		Specifies the address and length. The address of the data area alternatively refers to: • Bit memory address area • Data block area
NDR	O	BOOL		New data accepted
ERROR	O	BOOL		Error display
STATUS	O	WORD		Status display
LEN	O	INT		Number of bytes accepted into the data area from the Ethernet CP

When using the functions AG_SEND and AG_RECV, data is transported to the communication partner via the Ethernet bus of the CP. The communication partner is configured in STEP 7 in the "NetPro" tool.

The special feature when calling the functions involves the specification of parameter "LADDR" at the named blocks. In the case of SINUMERIK 840D sl, value W#16#8110 must be connected to parameter "LADDR".

In the basic program, this function is available under the FC number 1006 (this FC corresponds to the FC number FC 6 in the library "SIMATIC_NET_CP").

The block can also be used in a SIMATIC-CPU 3xx with CP343-1.

The protocols TCP and UDP are supported. TCP is the preferred protocol.

12.16 Signal/data descriptions

12.16.1 Interface signals NCK/PLC, HMI/PLC, MCP/PLC

References

The NCK/PLC, HMI/PLC and MCP/PLC interface signals are contained in the Lists document.

Lists sl (Book2)

The reference code contained therein (according to the signal names) refer to the respective function description, in which the signal is described.

The NCK signals that are evaluated by the basic program and transferred in conditioned form to the user interface are presented in the following sections.

12.16.2 Decoded M signals

The M functions programmed in the part program, ASUB or synchronized actions are channel specifically transferred from the NC to the PLC:

- M functions from channel 1: DB 21
- M functions from channel 2: DB 22
- etc.

The signal length is one PLC cycle.

Note

The spindle-specific M functions below are not decoded: M3, M4, M5, and M70.

Address in DB 21, ...	Variable	Type	Comment
DBX 194.0 ... 7	M_Fkt_M0 ... M7	BOOL	M signals M0 ... M7
DBX 195.0 ... 7	M_Fkt_M8 ... M15	BOOL	M signals M8 ... M15
DBX 196.0 ... 7	M_Fkt_M16 ... M23	BOOL	M signals M16 ... M23
DBX 197.0 ... 7	M_Fkt_M24 ... M31	BOOL	M signals M24 ... M31
DBX 198.0 ... 7	M_Fkt_M32 ... M39	BOOL	M signals M32 ... M39
DBX 199.0 ... 7	M_Fkt_M40 ... M47	BOOL	M signals M40 ... M47
DBX 200.0 ... 7	M_Fkt_M48 ... M55	BOOL	M signals M48 ... M55
DBX 201.0 ... 7	M_Fkt_M56 ... M63	BOOL	M signals M56 ... M63
DBX 202.0 ... 7	M_Fkt_M64 ... M71	BOOL	M signals M64 ... M71
DBX 203.0 ... 7	M_Fkt_M72 ... M79	BOOL	M signals M72 ... M79
DBX 204.0 ... 7	M_Fkt_M80 ... M87	BOOL	M signals M80 ... M87
DBX 205.0 ... 7	M_Fkt_M88 ... M95	BOOL	M signals M88 ... M95
DBX 206.0 ... 3	M_Fkt_M96 ... M99	BOOL	M signals M96 ... M99

Note

The M02/M30 auxiliary function output to the PLC does not state that the part program has been terminated. To determine definitely the end of a part program in the channel, the following interface signal must be evaluated:

DB21, ...DBX33.5 (M02/M30 active)

The channel status must be RESET. The auxiliary function output could arise from an asynchronous subroutine (ASUB) or a synchronized action and has nothing to do with the real end of the parts program in this case.

12.16.3 G Functions

The M functions programmed in the part program, ASUB or synchronized actions are channel specifically transferred from the NC to the PLC:

- G functions from channel 1: DB 21
- G functions from channel 2: DB 22
- etc.

The signal length is one PLC cycle.

POWER ON

After POWER ON, the value zero, i.e. active G groups undefined, is specified in the NC/PLC interface for all G groups.

Part program end or abort

After part program end or abort, the last state of the G group is retained.

NC START

After NC-START the values of the 8 G-groups specified in the machine data:
MD22510 $NC_ GCODE_GROUPS_TO_PLC
are overwritten according to the default setting set via the machine data as well as the values programmed in the part program.

Address in DB 21, ...	Variables	Type	Basic position	Comment
DBB 208	G_FKT_GR_1	BYTE	0	Active G function of group 1
DBB 209	G_FKT_GR_2	BYTE	0	Active G function of group 2
DBB 210	G_FKT_GR_3	BYTE	0	Active G function of group 3
DBB 211	G_FKT_GR_4	BYTE	0	Active G function of group 4
DBB 212	G_FKT_GR_5	BYTE	0	Active G function of group 5
DBB 213	G_FKT_GR_6	BYTE	0	Active G function of group 6
DBB 214	G_FKT_GR_7	BYTE	0	Active G function of group 7
DBB 215	G_FKT_GR_8	BYTE	0	Active G function of group 8
DBB 216	G_FKT_GR_9	BYTE	0	Active G function of group 9
DBB 217	G_FKT_GR_10	BYTE	0	Active G function of group 10
DBB 218	G_FKT_GR_11	BYTE	0	Active G function of group 11
DBB 219	G_FKT_GR_12	BYTE	0	Active G function of group 12
DBB 220	G_FKT_GR_13	BYTE	0	Active G function of group 13
DBB 221	G_FKT_GR_14	BYTE	0	Active G function of group 14
DBB 222	G_FKT_GR_15	BYTE	0	Active G function of group 15
DBB 223	G_FKT_GR_16	BYTE	0	Active G function of group 16
DBB 224	G_FKT_GR_17	BYTE	0	Active G function of group 17
DBB 225	G_FKT_GR_18	BYTE	0	Active G function of group 18
DBB 226	G_FKT_GR_19	BYTE	0	Active G function of group 19
DBB 227	G_FKT_GR_20	BYTE	0	Active G function of group 20
DBB 228	G_FKT_GR_21	BYTE	0	Active G function of group 21
DBB 229	G_FKT_GR_22	BYTE	0	Active G function of group 22
DBB 230	G_FKT_GR_23	BYTE	0	Active G function of group 23
DBB 231	G_FKT_GR_24	BYTE	0	Active G function of group 24

Address in DB 21, ...	Variables	Type	Basic position	Comment
DBB 232	G_FKT_GR_25	BYTE	0	Active G function of group 25
DBB 233	G_FKT_GR_26	BYTE	0	Active G function of group 26
DBB 234	G_FKT_GR_27	BYTE	0	Active G function of group 27
DBB 235	G_FKT_GR_28	BYTE	0	Active G function of group 28
DBB 236	G_FKT_GR_29	BYTE	0	Active G function of group 29
DBB 237	G_FKT_GR_30	BYTE	0	Active G function of group 30
DBB 238	G_FKT_GR_31	BYTE	0	Active G function of group 31
DBB 239	G_FKT_GR_32	BYTE	0	Active G function of group 32
DBB 240	G_FKT_GR_33	BYTE	0	Active G function of group 33
DBB 241	G_FKT_GR_34	BYTE	0	Active G function of group 34
DBB 242	G_FKT_GR_35	BYTE	0	Active G function of group 35
DBB 243	G_FKT_GR_36	BYTE	0	Active G function of group 36
DBB 244	G_FKT_GR_37	BYTE	0	Active G function of group 37
DBB 245	G_FKT_GR_38	BYTE	0	Active G function of group 38
DBB 246	G_FKT_GR_39	BYTE	0	Active G function of group 39
...	...	...	...	...
DBB 271	G_FKT_GR_64	BYTE	0	Active G function of group 64

A complete listing of all the G functions is given in:
References:
Programming Manual Fundamentals; Chapter: "List of G-Functions/Preparatory functions"

12.16.4 Message signals in DB 2

DB 2 allows the user to display the messages for individual signals on the operator panel. As the lists of interface signals show, signals are divided into predefined groups. When a message occurs, disappears or is acknowledged, the number entered in the message number column is transferred to the HMI. Text can be stored in the HMI for each message number.

References:

- Lists sl (Book2), see Section "PLC-Messages (DB 2)".
- Startup manual; Chapter "Alarm and message texts"

Note

The number of user areas can be parameterized via FB 1.

After the configuration has been modified (FB 1: MsgUser), DB 2/3 must be deleted.

12.17 Programming tips with STEP 7

Some useful tips on programming complex machining sequences in STEP7 are given below. This is essentially handling of the data type POINTER or ANY.

Fundamental tips on the structure of the data type POINTER and ANY see:
References:
STEP 7-Manual; Chapter: "Designing user programs" > "Register of CPU and saving of data"

12.17.1 Copying data

Copying variants

For the high-speed copying of data from one DB into another it is recommended

- **for larger data quantities** to use the system function SFC BLKMOV or SFC FILL, because here a high-speed copying takes place.
- the routine given below is **for smaller data quantities**, because the supply of ANY parameter to the SFCs consumes additional time.

Example

```
Code                                    Comment
                                        // DB xx.[AR1] is the source
                                        // DI yy.[AR2] is the destination
OPEN        DB 100;                     //Source DB
LAR1        P#20.0;                     //Source start address on data byte 20
OPEN        DI 101;                     //Destination DB
LAR2        P#50.0;                     //Destination start address on data byte 50
                                        //AR1, AR2, DB, DI loaded beforehand
L           4;                          //Transfer 8 bytes
M001:
L           DBW [AR1,P#0.0];            //Copy word-oriented
T           DIW [AR2,P#0.0];
+AR1        P#2.0;
+AR2        P#2.0;
TAK;
LOOP        M001;
```

12.17.2 ANY and POINTER

The following programming examples show the programming mechanism. They demonstrate how input/output and transit variables (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT) are accessed by data types "POINTER" or "ANY" within an FC or FB. The access operations are described in such a way that a part symbolic method of programming can be used.

12.17.2.1 Use of POINTER and ANY in FC

Function

FC 99 has inputs parameters that are defined as POINTER or ANY.

The example shows a body program via which the subcomponents of the POINTER or ANY can be accessed. In this case, the DB parameterized with POINTER or ANY is opened and the address offset stored as a crossarea pointer in address register AR1, Thus allowing access to data elements of variables (generally structures and arrays) that are addressed via the POINTER, ANY.

This access operation is described at the end of the relevant program sequence in the example. With data type ANY, it is also possible to execute a check or branch when the variable is accessed based on the data type and the number of elements involved.

Example

```
Code                                          Comment
FUNCTION FC 99: VOID
VAR_INPUT
    Row : BYTE ;
    Convert : BOOL ;                          //Activate numerical conversion
    Addr: POINTER;                            //Points to variable
    Addr1 : ANY ;
END_VAR
VAR_TEMP
    dbchr : WORD ;
    Number: WORD ;
    type : BYTE ;
END_VAR
BEGIN
```

```
Code                                                Comment
NETWORK
TITLE =
                                                    //POINTER
    L               P##Addr;
    LAR1 ;                                          //Retrieve pointer
    L               W [AR1,P#0.0];                  //Retrieve DB number
    T               #dbchr;
    L               D [AR1,P#2.0];                  //Offset part of pointer
LAR1 ;
AUF DB [#dbchr];                                    //Open DB of variables
L B [AR1,P#40.0];                                   //Retrieve byte value using pointer with
                                                    //address offset 40
                                                    //ANY
    L               P##Addr1;
    LAR1 ;                                          //Retrieve ANY
    L               B [AR1,P#1.0];                  //Retrieve type
    T               #typ;
    L               W [AR1,P#2.0];                  //Retrieve amount
    T               #Amount;
    L               W [AR1,P#4.0];                  //Retrieve DB number
    T               #dbchr;
    L               D [AR1,P#6.0];                  //Offset part of pointer
LAR1 ;
    OPEN            DB [#dbchr];                    //Open DB of variables
    L               B [AR1,P#0.0];                  //Retrieve byte value using ANY
```

12.17.2.2 Use of POINTER and ANY in FB

Function

FB 99 has inputs parameters that are defined as POINTER or ANY.

The example shows a body program via which the subcomponents of the POINTER or ANY can be accessed. In this case, the DB parameterized with POINTER or ANY is opened and the address offset stored as a crossarea pointer in address register AR1, thus allowing access to data elements of variables (generally structures and arrays) that are addressed via the POINTER, ANY.

This access operation is described at the end of the relevant program sequence in the example. With data type ANY, it is also possible to execute a check or branch when the variable is accessed based on the data type and the number of elements involved.

Example

```
Code                                                    Comment
FUNCTIONBLOCK FB 99
VAR_INPUT
    Row : BYTE ;
    Convert : BOOL ;                                    //Activate numerical conversion
    Addr: POINTER;                                      //Points to variable
    Addr1 : ANY ;
END_VAR
VAR_TEMP
    dbchr : WORD ;
    Number: WORD ;
    type : BYTE ;
END_VAR
BEGIN
NETWORK
TITLE =
                                                        //POINTER
    L                   P##Addr;
    LAR1 ;                                              //Retrieve pointer from instance DB
    L                   DIW [AR1,P#0.0];                //Retrieve DB number
    T                   #dbchr;
    L                   DID [AR1,P#2.0];                //Offset part of pointer
    LAR1 ;
    OPEN                DB [#dbchr];                    //Open DB of variables
    L                   B [AR1,P#40.0];                 //Retrieve byte value using pointer with
                                                        //address offset 40
                                                        //ANY
    L                   P##Addr1;
    LAR1 ;                                              //Retrieve ANY from instance DB
    L                   DIB [AR1,P#1.0];                //Retrieve type
    T                   #typ;
    L                   DIW [AR1,P#2.0];                //Retrieve amount
    T                   #Amount;
    L                   DIW [AR1,P#4.0];                //Retrieve DB number
    T                   #dbchr;
    L                   DID [AR1,P#6.0];                //Offset part of pointer
    LAR1 ;
    OPEN                DB [#dbchr];                    //Open DB of variables
    L                   B [AR1,P#0.0];                  //Retrieve byte value using ANY
```

12.17.2.3 POINTER or ANY variable for transfer to FC or FB

POINTER or ANY variable

With version 1 or later of STEP 7 it is possible to define a pointer or ANY in VAR_TEMP.

The following two examples show how an ANY can be supplied.

Example 1: Transfer ANY parameter via a selection list to another FB (FC)

Several ANY parameters are defined in an FB (FC). A specific ANY parameter must now be chosen from a selection list for transfer to another FB (FC). This can only be done by means of an ANY in VAR_TEMP. 1 to 4 can be set in parameter "WhichAny" in order to select Addr1 to Addr4.

Note

Address register AR2 is used in the block. However, this address register AR2 is also used for multiinstance DBs. For this reason, this FB should **not** be declared as multi-instance DB.

```
Code                                    Comment
FUNCTIONBLOCK FB 100
CODE_VERSION1                           //starting from STEP 7 Version 2 for deactivating the
                                        //multi-instance DB
VAR_INPUT
WhichAny : INT ;
    Addr1 : ANY ;                       //Observe predetermined order
    Addr2 : ANY ;
    Addr3 : ANY ;
    Addr4 : ANY ;
END_VAR
VAR_TEMP
    dbchr : WORD ;
    Number: WORD ;
    type : BYTE ;
    Temp_addr : ANY ;
END_VAR
BEGIN
NETWORK
```

```
Code                                          Comment
TITLE =
L           WhichAny;
DEC 1;
L           P#10.0;                           //10 bytes per ANY
*I;
LAR2;
L           P##Addr1;
+AR2;                                         //Add ANY start addresses
L           P##Temp_addr;
LAR1 ;                                        //Retrieve pointer from VAR_TEMP
L           DID [AR2,P#0.0];                  //Transfer pointer value to VAR_TEM
T           LD [AR1,P#0.0];
L           DID [AR2,P#4.0];
T           LD [AR1,P#4.0];
L           DIW [AR2,P#8.0];
T           LW [AR1,P#8.0];

CALL FB 101, DB 100
            (ANYPAR := #Temp_addr);           //ANYPAR is data type ANY
```

Example 2: Transfer an ANY parameter constructed earlier to another FB (FC)

An ANY parameter that has already been compiled must be transferred to another FB (FC). This can be done only by means of an ANY stored in VAR_TEMP.

```
Code                                          Comment
FUNCTIONBLOCK FB 100
VAR_INPUT
    DBNumber: INT ;
    DBOffset : INT ;
    Data type: INT ;
    Number: INT ;
END_VAR
VAR_TEMP
    dbchr : WORD ;
    Temp_addr : ANY ;
END_VAR
BEGIN
NETWORK
TITLE =
L           P##Temp_addr;
LAR1 ;                                        //Retrieve pointer from VAR_TEMP
L           B#16#10;                          //ANY identifier
```

```
Code                                          Comment
T           LB [AR1,P#0.0];
L           Data type;
T           LB [AR1,P#1.0];
L           Amount;
T           LW [AR1,P#2.0];
L           DBNumber;
T           LW [AR1,P#4.0];
L           DBOffset;
SLD 3;                                        //Offset is a bit offset
T           LD [AR1,P#6.0];
CALL FB 101, DB 100
            (ANYPAR := #Temp_addr);           //ANYPAR is data type ANY
```

12.17.3 Multiinstance DB

Function

From Version 2 in STEP 7, you can provide multi-instance enabled FBs, i.e. with multi-instance DBs. The primary characteristic of multiinstance DBs is that a data module can be used for various instances of FBs (see STEP 7 documentation), The quantity structure of the DBs can be optimized this way.

Multi-instance DBs should be activated only when they are actually going to be used since they increase the runtime and code size of the FBs.

Note

When complex programs are implemented in multiinstance enabled FBs that use a pointer and address register, it is important for the programmer to observe certain rules.

With multiinstance DBs, the start address of the variable (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) is transferred with the DI data block register and address register AR2. When variables are accessed within the multiinstance enabled FB, the compiler independently controls the access operation via address register AR2. However, when complex program sections also have to work with address registers in the same FB (e.g. to copy data), then the old contents of AR2 must be saved before the register is changed. The contents of AR2 must be restored to their original state before an instance variable (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) is accessed. The AR2 register of the instance is to be saved most usefully in a local variable (VAR_TEMP).

The command "Load pointer to an instance variable" returns a pointer value from the start of the instance data. To be able to access this variable via a pointer, the offset stored in AR2 must be added.

Example

```
Code                                    Comment
FUNCTION_BLOCK FB 99
VAR_INPUT
    varin: INT ;
END_VAR
VAR
    variable1: ARRAY[[0..9] of INT;
    variable2: INT ;
END_VAR
BEGIN
L               P##variable1;           //Pointer at start of ARRAY
                                        //The value 8500 0010 is now in the accumulator
                                        //and a cross-area pointer is in the AR2
                                        //Pointer. If one is to work across areas
                                        //then, during the addition of these
                                        //two pointers, an area is to be disabled.
AD              DW#16#00FF_FFFF,        //Skipping of an area
LAR1                                    //Load into AR1
TAR2;
+AR1 AR2;                               //AR2 instance offset to be added
                                        //You can now indirectly access the ARRAY
                                        //of variable 1 via AR1.
L               DIW [AR1, P#0.0];       //E.g. access to the first element
END_FUNCTION_BLOCK
```

12.17.4 Strings

The STRING data type is required by certain services of the basic program. For this reason, some additional facts about the string structure and general handling procedures for parameter assignments are given below.

Structure of strings

A data of type STRING is generally stored (defined) in a data block. There are two methods of defining a string:

1. Only the data type STRING is assigned to a variable. The STEP7 compiler automatically generates a length of 254 characters.
2. Data type STRING is assigned to a variable together with a string length in square parenthesis (e.g. [32]). With this method, the STEP7 compiler generates a string length corresponding to the input.

Two bytes more than prescribed by the definition are always stored for variables of the STRING data type. The STEP 7 compiler stores the maximum possible number of characters in the 1st byte. The 2nd byte contains the number of characters actually used. Normally, the useful length of the assigned strings is stored by the STEP 7 compiler. The characters (1 byte per character) are then stored from the 3rd byte onwards.

String parameters are generally assigned to blocks of the basic program by means of a POINTER or ANY. Such assignments must generally by made using symbolic programming methods. The data block, which contains the parameterizing string, must be stored in the symbol list. The assignment to the basic program block is then made by means of the symbolic data block name followed by a full stop and the symbolic name of the string variable.

12.17.5 Determining offset addresses for data block structures

Function

Another task, which occurs frequently, is symbolic determination of an offset address within a structured DB, e.g. an ARRAY or STRUCTURE is stored somewhere within the DB. After loading the address register symbolically with the start address, you might like to access the individual elements of the ARRAY or STRUCTURE via an address register. One way of loading the address register symbolically is to use an FC whose input parameter is a pointer. The address of the ARRAY or STRUCTURE is then assigned symbolically to the input parameter of this FC in the program. The program code in the FC now determines the offset address from the input parameter, and passes the offset address in the address register (AR1) to the calling function. Symbolic addressing is thus possible even with indirect access.

Example

```
Code                                Comment
FUNCTION FC 99: VOID
VAR_INPUT
    Addr: POINTER;                  //Points to variable
END_VAR
BEGIN
NETWORK
TITLE =
L         P##Addr;
LAR1 ;                              //Retrieve pointer from Addr
L         D [AR1,P#2.0];            //Offset part of pointer of variable
LAR1 ;
END_FUNCTION
```

12.17.6 FB calls

Function

For optimizing the flow speeds, it is useful to call all function block calls with many static parameters, such as the blocks FB 2, 3, 4, 5, and 7 provided by the basic program in start-up with the related instance parameters. In the start-up (OB 100), the preassignment of the parameters must be done, which can then no longer be changed in the cyclic part (OB 1). These fixed parameter values are no longer parameterized in the cyclic call, because they have already been written in the Instance DB.

Example Parameterization of FB 2 with instance DB 110

The following example shows how a useful distribution in OB 100 and OB 1 portion is to be implemented.

First, the usual call in the cyclic program is displayed.

```
CALL FB 2, DB 110 (
      Req :          M 100.0,
      NumVar :       2,                          //Read 2 variables
      Addr1 :        NCVAR.C1_RP_rpa0_0
      Line1 :        W#16#1
      Addr2 :        NCVAR.C1_RP_rpa0_0
      Line2 .        W#16#2
      Error :        M1.0,
      NDR :          M1.1,
      State :        MW 2,
      RD1 :          P#M 4.0 REAL 1,
      RD2 :          P#M 24.0 REAL 1,
```

The modified version of the program call starts from here.
Here the call in OB 100 is displayed:

```
CALL FB 2, DB 110 (
      Req :          FALSE,
      NumVar :       2,                          //Read 2 variables
      Addr1 :        NCVAR.C1_RP_rpa0_0
      Line1 :        W#16#1
      Addr2 :        NCVAR.C1_RP_rpa0_0
      Line2 .        W#16#2
      RD1 :          P#M 4.0 REAL 1,
      RD2 :          P#M 24.0 REAL 1,
```

Here the call still remaining in OB 1 is displayed:

```
CALL FB 2, DB 110 (
      Req :=          M0.0,
      Error :=        M1.0,
      NDR :=          M1.1,
      State :=        MW 2,
```

Note

Owing to this measure, a shorter cycle time is achieved in OB 1, because the static parameter values need not be copied in the instance DB in each OB-1 cycle.

The savings of this variant:

The cyclic copying effort of 3 integer values and 4 ANY parameters with respect to the instance DB, which results from 3 time loading of a constant in the instance data block. In case of each ANY transfer, constants are loaded in the data block 4 times with subsequent transfer.

12.18 Data lists

12.18.1 Machine data

12.18.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10100	PLC_CYCLIC_TIMEOUT	Cyclic PLC monitoring time
14504	MAXNUM_USER_DATA_INT	Number of user data (INT)
14506	MAXNUM_USER_DATA_HEX	Number of user data (HEX)
14508	MAXNUM_USER_DATA_FLOAT	Number of user data (FLOAT)
14510	USER_DATA_INT	User data (INT)
14512	USER_DATA_HEX	User data (HEX)
14514	USER_DATA_FLOAT[n]	User data (FLOAT)

Note

Machine data in integer/hex format is operated in the NC as DWORD. A machine data in floating point format is managed in the NC as FLOAT (8-Byte IEEE) They are stored only in the NC/PLC interface and can be read by the PLC user program from DB 20 even during PLC booting.

12.18.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
28150	MM_NUM_VDIVAR_ELEMENTS	Number of elements for writing PLC variables

P4: PLC for SINUMERIK 828D 13

13.1 Overview

13.1.1 PLC firmware

The PLC of the SINUMERIK 828D is an integrated PLC based on the SIMATIC S7-200 command set.

The PLC user program is essentially programmed using a Windows PC with the "Programming Tool PLC828". In addition, the PLC can be diagnosed and edited via the operator interface of the control. A "Ladder-Add-On-Tool" is available in the control for this purpose.

Note the following special features:

- The PLC user program is completely programmed in ladder logic (LAD).
- A subset of the programming language of the S7-200 is supported.
- When loading to the CPU, in addition to the code for execution, the complete project data (including symbols and comments) is loaded into the control. This means that the control always has the project that matches the currently running PLC user program.
- When loading from the CPU, the complete project data (including symbols and comments) is loaded into the Programming Tool PLC828 and can be processed/edited using this.
- The user must manage the data and process information according to type. The declared data type must be used consistently each time that the data is accessed.

13.1.2 PLC user interface

The user interface is set-up by the PLC firmware, which also organizes the exchange of all signals and data between the PLC on one side and the NCK and HMI on the other side.

The user interface comprises the parts:

- Data interface with cyclic exchange (see "Data interface (Page 1258)")
- Function interface with function or task-related data exchange (see "Function interface (Page 1264)").

The structured data of these interfaces (retentive and non-retentive) are made available to the user by the firmware by assigning to data blocks: The NC (NCK, tool manager, NC channel, axes, spindles, …) and the HMI are "Communication partners" of the PLC user program.

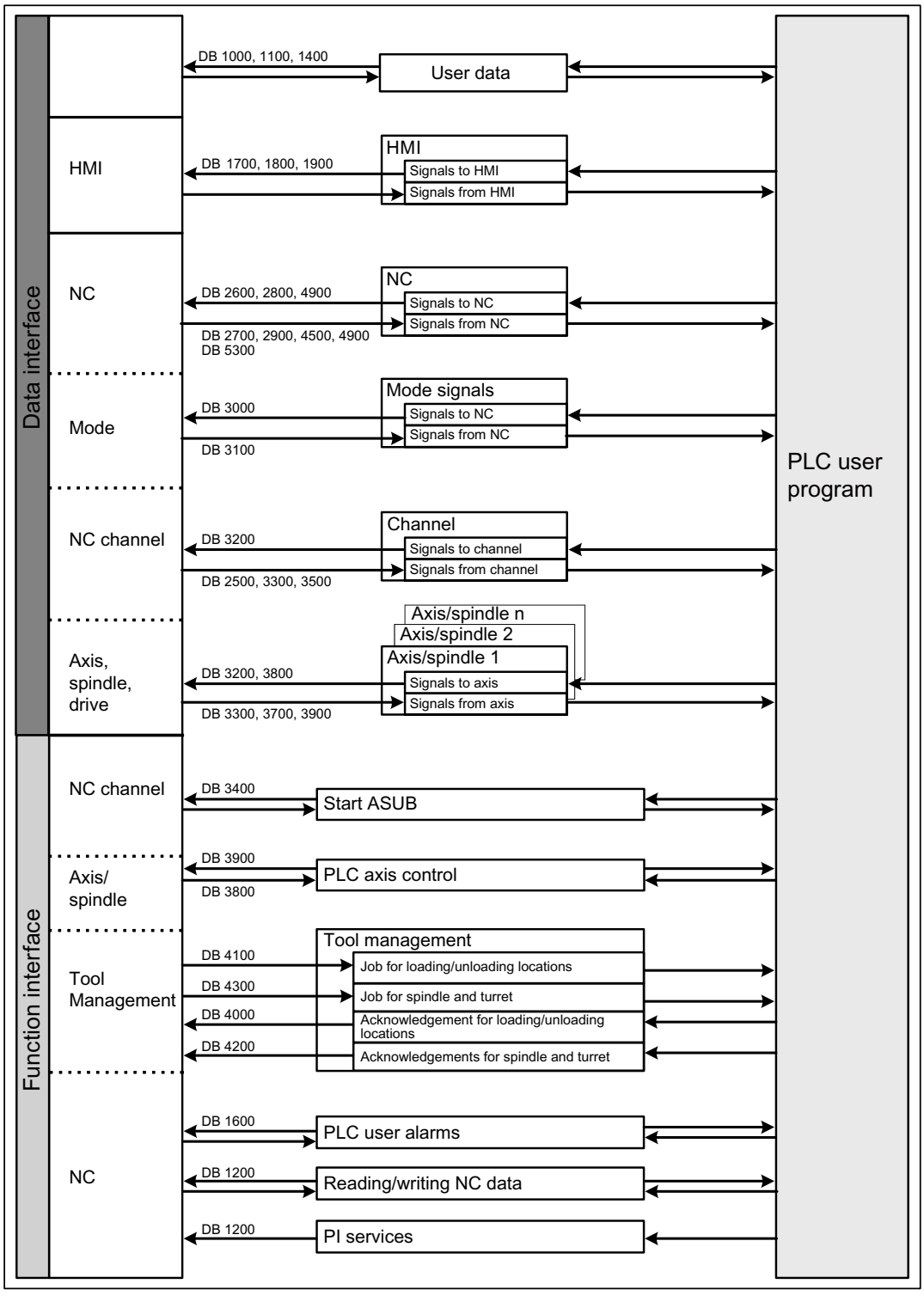


Figure 13-1 Overview of the user interface of the PLC 828D

13.1.2.1 Data that are cyclically exchanged

Data which is exchanged between the PLC and NC on one side as well as between the PLC and HMI on the other side.

Data **to** the PLC are provided by the firmware at the cyclic start of the user program. This ensures, for example, that the signals from the NCK remain constant throughout a cycle.

The firmware transfers data **from** the PLC to the NKC or HMI at the cycle end of the user program.

Interface PLC ↔ NCK

The cyclic data include, e.g. status signals ("program running", "program interrupted") and control signals (start, stop) and auxiliary and G functions.

Data are structured in signals for:

- Modes
- Channels
- Axes/spindles
- General NCK signals

Interface PLC ↔ HMI

These are signals for:

- Program selection via lists
- Messenger control command
- General signals from/to HMI
- Signals from/to the maintenance planner
- Signals from operator panel (retentive area)
- General selection/status signals from/to HMI (retentive area)

13.1.2.2 Alarms and messages

The user interface in DB1600 offers the option of displaying fault and operating messages on the HMI.

The firmware evaluates the signals that have been entered and sends these as coming and going alarms and messages to the HMI where they are displayed. The HMI manages the fault texts.

13.1.2.3 Retentive data

For the retentive data there are the user data blocks DB9000 - DB9063 and the data area DB1400.DBW0 - DBW127. There the user can store all data that should remain valid after POWER OFF/ON. The retentive data is stored in the non-volatile memory, however, not for data backup.

13.1.2.4 Non-retentive data

Non-retentive data (e.g. bit memories, timers and counters) are deleted every time the control is booted.

13.1.2.5 PLC machine data

The PLC machine data are in the NCK machine data area. At POWER ON, this data is transferred by the PLC firmware into DB4500 of the PLC user interface where it can be evaluated by the PLC user program.

References

SINUMERIK 828D Parameter Manual

13.1.3 PLC key data

The integrated PLC has a program memory of 24000 PLC operations which are completely executed in one fixed PLC cycle.

A maximum of 500 operations can be executed in the INT0 interrupt program that can be optionally used. It is executed servo-synchronous and allows the fastest possible reaction to process events. This is the reason that interrupt-capable PLC I/O modules are not required.

Data	Number	Special features
Main program (MAIN)	1	
Subprograms (SBRx)	256	
Interrupts	2	
Time controlled interrupt	1	Servo-synchronous interrupt program
Alarms/messages	248	
Bit memory	4096	Non-retentive
Counters	64	Non-retentive
Timers, of which:	128	Non-retentive
10 ms increment interval	112	
100 ms increment interval	16	
User data block each with a max. of 512 bytes	64	Address range DB9000 to DB9063
Data transmission NCK ↔ PLC		Via fixed parameterizable interface

13.1.4 PLC I/O, fast onboard inputs/outputs

For information on the properties of the rapid onboard inputs/outputs and their response times, see Section "Fast on-board inputs and outputs (Page 1199)".

For information on the I/O modules, the machine control panels as well as the assignment of the onboard inputs/outputs to the PLC, see:
References:
Manual PPU SINUMERIK 828D

13.1.5 PLC Toolbox

13.1.5.1 Star/delta changeover

For star/delta changeover, the following block is provided in the PLC Toolbox:

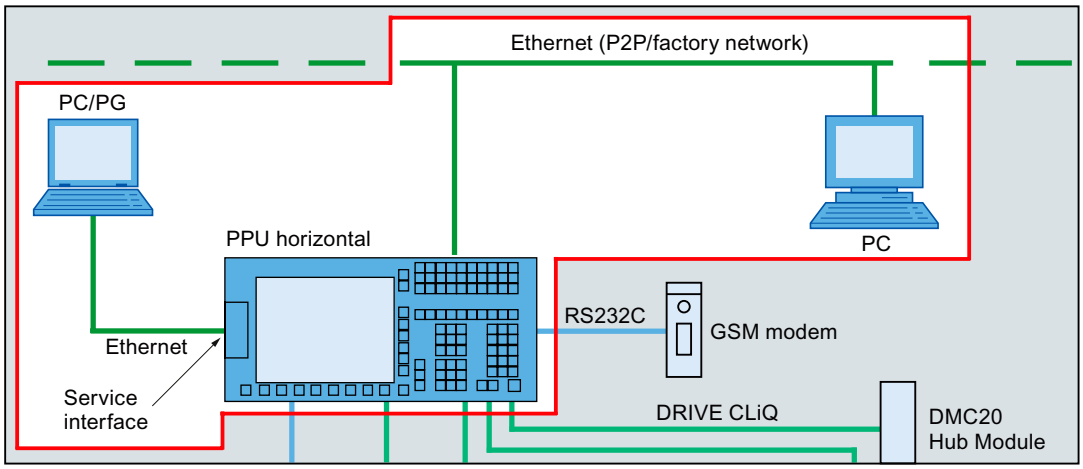
- StarDelta

Note

This block can be used to perform a star/delta changeover - also for 1PH8 spindle motors with SMI connected to a SINAMICS S120.

13.2 Programming Tool PLC828

The "Programming Tool PLC 828" is the tool with which PLC programs can be generated in a user-friendly fashion. This is a Windows program and must be installed on a Windows PC. This PC must be equipped with an Ethernet port for online access to the control and must be connected to the control via the factory network or the service interface.



References:
Commissioning Manual SINUMERIK 828D

When calling the Programming Tool PLC828 – without specifying an existing project – then implicitly a new project is created with the "Project1" default name. This project can be immediately used to generate the PLC user program and then saved under any name and loaded into the control system.

Existing projects can be opened in the typical Windows fashion, also typical for Windows, a user-friendly online help is available by pressing F1.

13.3 Programming

13.3.1 Introduction

13.3.1.1 Important terms

When calling the Programming Tool PLC828, a new project is implicitly created or a specified one is opened.

However, what is a project, a program, a data block? What are data classes?

The most important terms will now be briefly explained. For additional and more detailed information, see:

References:

- Programming Tool PLC828, online help
- S7-200 System Manual

Project

A project is the largest possible organizational unit for the user when working with the PLC. It is comparable with a container, and can:

- Accept program and data blocks, symbol and status tables, cross-references as well as interface and debug settings.
- Be saved on a data carrier or loaded from there.
- Be loaded into the CPU or retrieved from there.

Data classes

Data classes are especially the properties of actual values of such data blocks that the user explicitly brings into the project. (Data blocks that are inherent to the system are not meant, e.g. the user interface.)

The data classes "Manufacturer", "Individual" and "User" were introduced into SINUMERIK 828D in order to:

- Be able to assign the DB actual values to specific user groups.
- Be able to load values belonging to a user group (i.e. data class) into or out of the CPU.
- Simplify troubleshooting and maintenance.

All programs (with the exception of the two interrupt programs INT100 and INT101) and all data block structures (i.e. the inner structure, the "Type" of data blocks) and data block initial values have the "Manufacturer" data class.

The interrupt programs INT100 and INT101 are of the "Individual" type.

Program

A program (also a program organization unit "POU") is a block for a sequence of commands (including comments) that the users assemble one after the other using the LAD Editor to solve their particular task. The users have three types of these POUs at their disposal:

- The main program (MAIN)

  There is only one of these. The system calls this in the PLC cycle.

- 256 optional subprograms (SBR_xyz)

  Subprograms are used to structure and encapsulate functions.

  Once coded, they can be called a multiple number of times.

- Three optional interrupt programs

  These are executed with a higher priority at a different location in the PLC cycle and are reserved for special tasks.

Data block

A data block is a block for data (initial values, actual values) and comments with the following properties:

- The data is saved in precisely the same sequence as specified by the user. This means that the inner structure of the data block is defined and if several data blocks are created with the same inner structure (i.e. the same type), then a certain data is always located at a specific location. This location is called offset and is the relative length in bytes from the beginning of the data block (DB) up to the actual piece of data.
- Initial values can be assigned to the data that they assume after being loaded into the PLC for the first time.
- The actual values of the data can also be read online from the control, changed and also saved with the project.

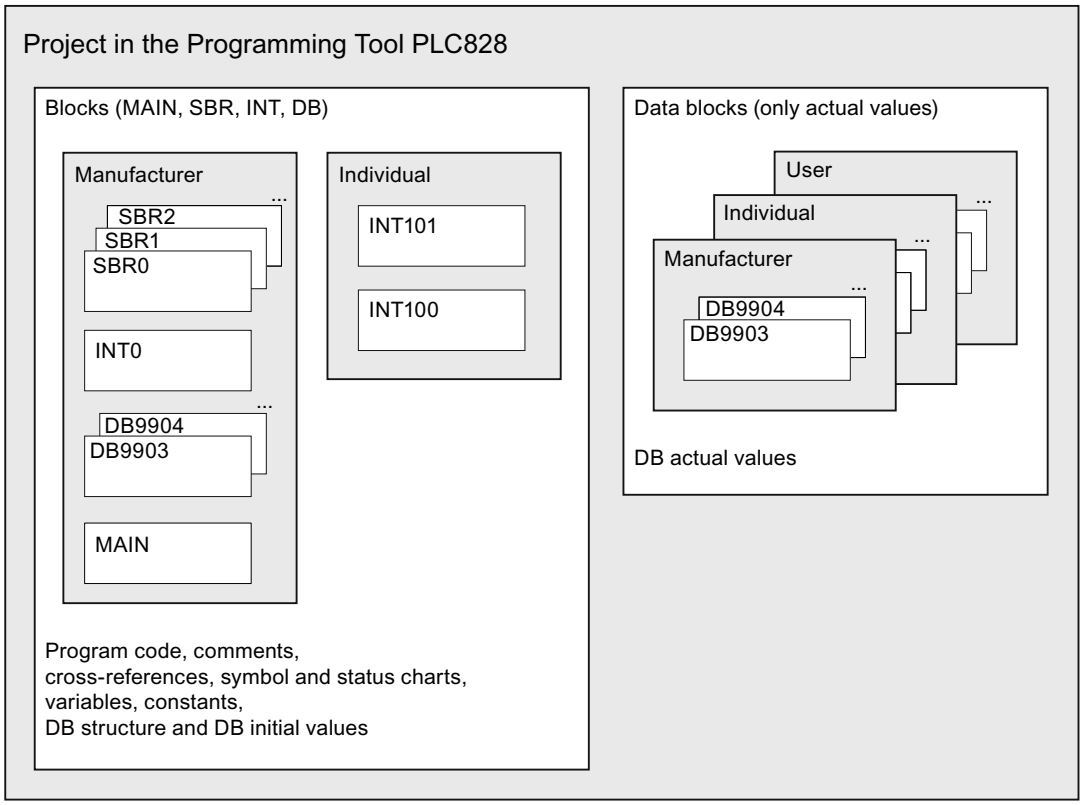


Figure 13-2 Example of a project structure

Symbol tables

The symbol table is used for symbolic addresses. Frequently, symbols simplify programming and increase the transparency of programs. In the compiled program, which is loaded into the target system, all of the symbols are converted into absolute addresses. The data relating to the symbol table is loaded into the target system.

Status table

In the status tables, you can observe how the process values change as the program is executed. Status tables are not loaded into the target system. They are only intended to be able to monitor the activities of the target system (or the simulated target system).

Cross references

The following are displayed in tabular form in the cross reference window:

- The symbolic or absolute addresses and the locations where they are used
- The bytes that are being used
- The bits that are being used

The cross references and the data on the elements used are not loaded into the target system.

13.3.1.2 Create/open a project

Creating a new project

- Double-click on the "Programming Tool PLC828" button.

or

- In the Windows start menu, select the command "Programming Tool PLC828" > "Programming Tool PLC828".

The Programming Tool PLC828 is started and a new project is opened.

Opening an existing project

- In the Programming Tool PLC828, the menu "File", select one of the following commands:
  - "Open"

    Navigate to an existing project and open it.
  - "File name"

    If you just recently worked in a project, this project is listed in the "File" menu and you can directly open it without first having to select it in the dialog box.
- You can also navigate to the required directory in Windows Explorer and directly open the project from there without having to first start the Programming Tool PLC 828. Your project is in a single file with the ***.ptp** extension.

Note

If you have created a project, then you can start to write your program. However, you should have executed the following tasks beforehand:

- **Range check according to the target system**

  You can select the target system type before you write your program so that the Programming Tool PLC828 can check the parameter range corresponding to the target system. (If you have selected a CPU type for your project, the operations that cannot be used for your target system are marked with a red x in the operation tree.)
- **Setting-up the work environment**

  You can set up your work environment in various ways ("Windows look and feel"). More detailed information is provided in the online help under "Setting-up the Programming Tool display".

13.3.1.3 Program organization using the the Programming Tool

The Programming Tool PLC828 organizes your program in the Program Editor in individual tabs per POU. As standard, when creating a new project, the main program MAIN and a subprogram SBR_0 are created. MAIN is always in the first tab, all subprograms and interrupt programs that you have generated then follow.

MAIN

MAIN cannot be renamed and is permanently assigned to the "Manufacturer" data class.

Subprograms

Subprograms are useful if you wish to execute a function a multiple number of times. In order that you do not have to include the logic in the main program at each location where you wish to execute the function, you write the logic once in a subprogram and you then call this subprogram as often as required while executing the main program. Meaningful names can be assigned to subprograms; these are also assigned to the "Manufacturer" data class.

Advantages:

- The functional sequence in the main program is very transparent.
- Subprograms are easy to port. You can easily demarcate a function and without many resources, call it from other programs with other parameter values.

Note

The use of global variables restrict the portability of subprograms as the assignment of addresses in the variable memory of a program can always be in conflict with the assignment of addresses in another program. Subprograms, which access local variables for all address assignments, are on the other hand extremely easy to port, because there is no danger of conflicting addresses.

Interrupt programs

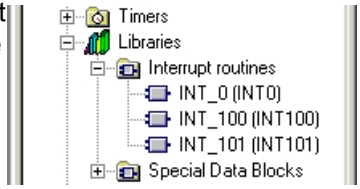
Interrupt programs are used to be able to handle special process conditions or requirements. The PLC828 makes a differentiation as follows:

Condition	Program	Description
Shortest response time: Executed in the **servo cycle**	INT_0	For extremely fast responses to events, which absolutely required this. In INT_0, the direct access commands, that either access an internal, I/O image updated in the servo cycle or only the hardware of the onboard IO, can be practically used. Data class: "Manufacturer" **Note:** The size of block INT_0 is limited to 500 operations.
Executed **before** MAIN, after reading the inputs	INT_100	Especially to marshal data that is read accessed in the additional cycle. Data class: "Individual"
Executed **after** MAIN, before writing to the outputs	INT_101	Especially when marshaling data that were previously written in the PLC cycle, but which should be output to other or additional outputs. Data class: "Individual"

Interrupt programs cannot be renamed, but they can be assigned to data classes.

The interrupt program is inserted in the project by double clicking on the corresponding name in the operation tree: "Libraries" > "Interrupt programs".

The PLC firmware then automatically makes the call.



Exiting program organizational units

As a result of the described program organization (each POU occupies its own tab), it is quite clear where MAIN and the individual sub and interrupt programs end; this is the reason that they do not require any special end delimiter.

See also

Addressing (Page 1204)

Type of memory (Page 1201)

13.3.1.4 Fast on-board inputs and outputs

The PLC of the PPU module can use eight rapid inputs and six rapid outputs that are provided "onboard" via the connectors X242 (DIN1-DIN4, DOUT1-DOUT4) and X252 (DIN9-DIN12, DOUT9-DOUT10). They therefore do not require any I/O modules and can be processed extremely fast. Per default, these inputs/outputs are assigned to the NC. The assignment to the NC or PLC is carried out via the following machine data:

MD10366 $MN_HW_ASSIGN_DIG_FASTIN[<n>]
(HW assignment for external digital inputs)

MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[<n>]
(HW assignment for external digital outputs)

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

NC/PLC assignment

Plug-in connector	Machine data	I/O assignment
X242	MD10366[0] = MD10368[0] = 10101_H	DIN1 - DIN4 and DOUT1 - DOUT4 to NC
	MD10366[0] = MD10368[0] = 10001_H	DIN1 - DIN4 and DOUT1 - DOUT4 to PLC
X252	MD10366[1] = MD10368[1] = 10101_H	DIN9 - DIN12 and DOUT9 - DOUT10 to NC
	MD10366[1] = MD10368[1] = 10001_H	DIN9 - DIN12 and DOUT9 - DOUT10 to PLC

Addressing

Addressing:	Address
On-board **in**puts:	`I256.0 ... I256.3`
On-board **out**puts:	`Q256.0 ... Q256.3`

Access

These inputs/outputs not assigned an image memory; reading and writing is performed directly from or to the hardware. For the fastest possible access (servo-synchronous), use of the direct operation commands in the interrupt program **INT0** is recommended:

Command	Symbol
Direct NO contact	-| I |-
Direct NC contact	-| /I |-
Directly set bit value	-(SI)-
Directly reset bit value	-(RI)-
Directly assign bit value	-(I)-

Response times

This results in the following response times depending on the position control cycle, place of execution and used digital inputs/outputs:

Position control cycle clock		1.5 ms	3 ms
	Processing via:	Response time [1]	
	Subprogram SBRx in cyclic operation (MAIN, OB1), digital inputs/outputs: I/O module PP 72/48...	12.5 ms	14 ms
	Interrupt program INT0 (servo-synchronous), digital inputs/outputs: On-board inputs/outputs of the PPU	3 ms	4.5 ms
[1] Signal at the input terminal → processing in the PLC program → signal at the output terminal			

13.3.2 Target system memory

13.3.2.1 Type of memory

Type	Description	Access in the bit format	Access in the type format	Access in the word format	Access in the double word format	can be retentive
I	Digital inputs and process image input	read / write	read / write	read / write	read / write	no
Q	Digital outputs and process image output	read / write	read / write	read / write	read / write	no
M	Internal bit memory	read / write	read / write	read / write	read / write	no
SM	Special bit memory (SM0.0 - SM0.6 are write-protected)	read / write	read / write	read / write	read / write	no
V	Variable memory	read / write	read / write	read / write	read / write	yes
T	Actual values of timers and time bits	Time bit read / write	no	actual value of time read / write	no	no
C	Actual values of the counters and count bits	Counter bit read / write	no	actual value of the counter read / write	no	no
AC	Accumulators	no	read / write	read / write	read / write	no
L	Local data memory	read / write	read / write	read / write	read / write	no

13.3.2.2 Addressing range of the target system

Address range of a memory type is the addressing range for this type of memory that spans the lowest and highest possible address numbers. The lowest and highest possible address number depends on the particular memory type.

If you execute operations which communicate with the target system, then the Programming Tool PLC828 recognizes your CPU version. When generating your program, you must ensure that you only use the address range valid for this CPU. If you attempt to load a program that accesses addresses ranges which are not valid for this CPU, then an error message is output.

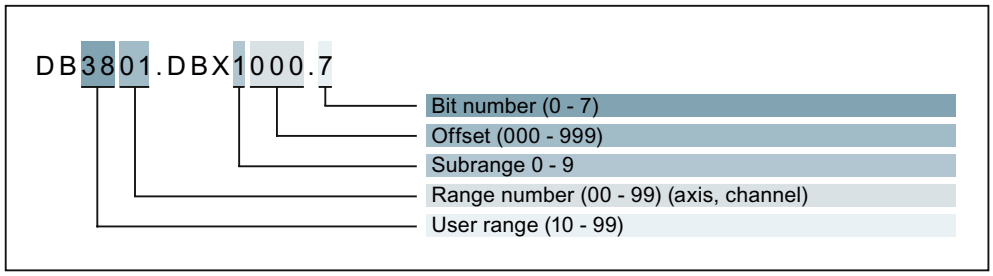
Address ranges

Access method	Type of memory	Min. and max. address numbers
Bit (byte.bit)	I	0.0 - 255.7 [1] 256.0 - 256.3 [2]
	Q	0.0 - 255.7 [1] 256.0 - 256.3 [2]
	M	0.0 - 999.7
	SM	0.0 - 0.6
	T	0 - 15 (100 ms) 16 – 127 (10 ms)
	C	0 – 63
	L	0.0 - 59.7
Byte	IB	0 - 255 [1]
	QB	0 - 255 [1]
	MB	0 – 999
	SMB	0
	LB	0 - 59
	AC	0 - 3
Word	IW	0 - 254 [1]
	QW	0 - 254 [1]
	MW	0 – 998
	T	0 - 15 (100 ms) 16 - 127 (10 ms)
	C	0 – 63
	LW	0 – 58
	AC	0 – 3
Double word	ID	0 - 252 [1]
	QD	0 - 252 [1]
	MD	0 996
	LD	0 - 56
	AC	0 - 3
User interface	DB	1000 - 7999 [3]
User data blocks	DB	9000 - 9063 [3]
Special data blocks	DB	9900 - 9999 [3]

1) Range of the process image variables! The assignment of these variables to the physical inputs and outputs can be taken from the system overview in the SINUMERIK 828D Commissioning Manual.

2) These addresses directly serve (i.e. without image memory) the eight on-board inputs and the six on-board outputs if these are assigned to the PLC.

3) Only the DB numbers are specified in the table for reasons of simplicity. Their addressing depends on the DB structure and is realized according to the following scheme:



Access	Example	Explanation
Bit	DB3801.DBX1000.7	Bit 7 of the byte with offset 0 in subrange 1 for axis 2 in user range 38
Byte	DB3801.DBB0	Byte with offset 0 in subrange 0 for axis 2 in user range 38
Word	DB4500.DBW2	Word with offset 2 in subrange 0 in range 0 in user range 45
Double Word	DB2500.DBD3004	Double word with offset 4 in subrange 3 in range 0 in user range 25

The access type is part of the address notation and must not be considered to be the same as the data type or confused with this (see "Data types (Page 1207)").

Note

The permitted **offset** for an address is dependent on the access:

- Bit or byte access: Any offset permitted.
  Byte-size variables are placed one beside another seamlessly in a DB.
- Word access: Offset must be divisible by 2.
  Word variables (2 bytes) are always saved on even offsets.
- Double word access: Offset must be divisible by 4.
  Double word variables (4 bytes) are always saved on offsets which are divisible by 4.

See also "Data types (Page 1207)".

User interface is an interface comprising data blocks that the firmware creates on the target system. It is used to exchange data between the PLC on one side and the NCK and HMI on the other side (→ List Manual SINUMERIK 828D).

This DB neither has to be loaded from the CPU nor into the CPU because it is created by the firmware and therefore belongs to the system.

User data blocks are exclusively created by the user. If it is intended to indirectly access blocks of the same structure, then these should be located one after the other regarding the numbering.

Special data blocks are permanently specified as far as their structure is concerned from the system and can be found in the Programming Tool PLC828 in "Libraries". However, whether they are integrated in the user program, assigned to a data class and loaded into the CPU is the responsibility of the users and their design philosophy (if e.g. a maintenance planner is not used, then these do not have to be integrated as far as the DB is concerned).

13.3.2.3 Addressing

Direct addressing

For direct addressing, the *memory type* as well as the *address number* must be specified (also refer to "Addressing range of the target system (Page 1201)").

The CPU memory (V, I, Q, M and SM) can be accessed bit by bit as well as in the byte, word and double-word format. The direct address comprises the memory type and a valid address number.

To access a bit in a memory area, enter the byte address and the number of the bits. Separate the byte from the bit using a decimal point.

Examples:

DB9900.DBX20.0	Bit 0 in byte 20 of DB9900
MB21,	Flag byte 21
QD16,	Output double word 16
I1.7,	Bit 7 in input byte 1

Indirect addressing

Indirect addressing can only be used for data blocks with the same structure (the same type). These are used so that when accessing data blocks, the number of the data block can be variably addressed. The data block number must be located in one of the accumulators AC0 ... AC3 (see "Data blocks (Page 1211)").

Absolute and symbolic addressing

You can specify the addresses in the operations in your program either absolutely or symbolically.

An *absolute address* specifies the memory type and the address number.

A *symbolic address* (briefly: symbol) specifies the address using a name (a combination of alphanumeric characters).

Global symbols are assigned in the symbol tables to their absolute address values and are valid throughout the project (global). This assignment can be made at any time.

Local symbols are assigned in the local variable table of the particular program and are only valid in this program (local).

Examples for displaying addresses in the program editor:

I0.0	An absolute address specifies the memory type and address number.
#Input1	The # character is located in front of a local symbol.
INPUT1	Global symbol
??.? or ????	Red question marks designate an address that has still not been defined (you must specify the address before you can compile the program).

Both addressing types – absolute and symbolic – are coupled to the particular view:

Menu "View" > "Symbolic addressing (Ctrl +Y)"

This setting should always be set using the menu "View > "Symbol table (Ctrl +T)".

Recommendation: Decide on one of the two addressing types and then keep this.

Global and local range of validity

Symbolic addresses that were assigned in the symbol table are globally valid. Symbolic addresses that were assigned in a local variable table are locally valid.

Local variables

Local variables are assigned in the local variable table of the particular POU and are limited to the validity range of the POU in which they were set-up. Each program organizational unit has its own local variable table.

Example:

You define a variable with the name INPUT1 in the local variable table of a subprogram with the SBR1 name.

If you refer to INPUT1 from SBR1, then the program editor identifies INPUT1 as a local variable from SBR1.

However, if you refer to INPUT1 at another location in the program (e.g. in MAIN or in a second program), the program editor does not recognize INPUT1 as a local variable and treats INPUT1 as a non-defined global symbol.

Note

Assigning names to local and global symbols

If you use the same name for an address at the local and at the global level, then the local use has priority. This means that if the program editor defines a definition for the name in the local variable table of a certain program block, then this definition is used. If a definition is not found, then the program editor checks the symbol table.

Example:

You define the global symbol "PumpOn" You also define "PumpOn" as local variable in SBR2, however, not in SBR1.

If the program is compiled, the local definition for "PumpOn" in SBR2 is used. The global definition is used for "PumpOn" in SBR1.

Note

Using local and global symbols

Local variables use the temporary, local memory of the target system. Subprograms that only use local variables and transfer parameters, are easy to port and can be flexibly used.

If you wish to use a parameter in several program organizational units, then it makes sense to define this parameter as global symbol in the symbol table - and not in the local variable table. This reason for this is that then you would have to include the parameter in every local variable table of the individual POUs.

Note

Initialization of local variables

As local variables occupy temporary memories, you must initialize the local variables in the POU each time that the POU is called. You **cannot** assume that a local variable keeps a data value from one POU call to the next.

13.3.2.4 Data types

When defining symbols in the global symbol table, a data type does not have to be explicitly specified, as it is implicitly specified by the data associated with the symbol.

If you assign values in the local variable table, then you must specify a data type for every local variable.

By explicitly specifying a data type for a value, you give the Programming Tool PLC828 clear instructions how much memory must be assigned for the value (e.g. the value 100 can be stored as BYTE, WORD or DWORD) and how the value is to be represented (e.g. should 0 be interpreted as BOOL or as numerical value?).

The operations and parameterized subprograms are recognized using a precise definition. This definition is also called signature. For all standardized operations, the data types permissible for the addresses of the operation are specified in the signature. For parameterized subprograms, the signature of the subprogram is generated by the user in the local variable table.

Data type check

The Programming Tool PLC828 offers a simple data type check. If a data type is specified for a local or global variable, the software checks that the data type of the address corresponds to the signature of the operation.

Elementary data types	Description	Memory area
BOOL (bit)	Boolean	0 ... 1
BYTE	Byte, unsigned	0 ... 255
WORD	Integer number (16 bit)	-32768 ... +32767
DWORD (Double Word)	Integer number (32 bit)	-2147483648 ... +2147483647
REAL	32-bit floating point	+/- 10^{-37} ... +/- 10^{+38}

Complex data types	Description	Memory area
TON	Switch-on delay	100 ms T0 ... T15 10 ms from T16
TOF	Switch-off delay	100 ms T0 ... T15 10 ms from T16
TONR	Switch-on delay, latching	100 ms T0 ... T15 10 ms from T16
CTU	Up counter	C0 ... C63
CTD	Down counter	C0 ... C63
CTUD	Up-down counter	C0 ... C63

The Programming Tool PLC828 has two data type check stages:

1. Elementary data type check

For the elementary data type check, if a symbol or a variable is assigned a data type, then all data types are automatically assigned, which correspond to the bit size of the user-defined data type. If, for instance, you specify DINT as data type, then the local variable is also automatically assigned the DWORD data type, because both data types are 32-bit types. The REAL data type is not automatically assigned although it also involves a 32-bit data type. The REAL data type is defined so that it does not have any equivalent data types: it is always clear. The elementary data type check is only performed when using local variables.

User-defined data types	Equivalent data type
BOOL	BOOL
BYTE	BYTE
WORD	WORD, INT
INT	WORD, INT
DWORD	DWORD, INT
DINT	DWORD, DINT
REAL	REAL

2. No data check

This mode is only available for global variables where no data types can be specified. If a data type check is not active, all data types of the same size are automatically assigned to the symbol.

Example:

A symbol, which is assigned to address DB1400.DBD4, then the following data types are automatically assigned by the programming software: DWORD, DINT and REAL.

User-defined address	Assigned equivalent data type
DB1400.DBX0.0	BOOL
DB1400.DBB0	BYTE
DB1400.DBW2	WORD, INT
DB1400.DBD4	DWORD, DINT, REAL

Advantages of the data type check

The data type check helps you to avoid programming errors that have been widely propagated through the program. If an operation supports numbers with signs, the Programming Tool PLC828 designates the use of unsigned numbers in addresses of operations.

Example:

The comparison < I is an operation with sign. -1 is less than 0 for addresses with sign. However, if the operation < I supports unsigned data types, then the programming itself must ensure that the following does not occur: While a program is being executed, an unsigned value of 40.000 is actually smaller than 0 for the operation < I. If it cannot be guaranteed that the unsigned numbers for signed operations do not exceed the positive and negative limit values, then unpredictable events can occur in your program or in the mode of operation of the control.

Working with operations to convert the data type

Conversion operations convert one data type into another. The Programming Tool PLC828 supports the following conversion operations to transfer values between the elementary data types.

Conversion of data types	Conversion operations	Complete data type check, permissible addresses	Data check, permissible addresses
INT in BCD	I_BCD	IN: INT OUT: INT	IN: WORD, INT OUT: WORD, INT
BCD in INT	BCD_I	IN: INT OUT: INT	IN: WORD, INT OUT: WORD, INT
DINT in REAL	DI_R	IN. DINT OUT: REAL	IN: DWORD, DINT OUT: REAL
REAL in DINT (ROUND)	TRUNC	IN: REAL OUT: DINT	IN: REAL OUT: DWORD, DINT

13.3.2.5 Constants

Range of constants

	Range without sign		Range with sign	
Size of the data	**Decimal:**	**Hexadecimal:**	**Decimal:**	**Hexadecimal:**
B (Byte)	0 ... 255	0 ... FF	-128 ... +127	80 ... 7F
W (Word)	0 ... 65535	0 ... FFFF	-32768 ... +32767	8000 ... 7FFF
D (Double word)	0 ... 4294967295	0 ... FFFF FFFF	-2147483648 ... +2147483647	8000 0000 ... 7FFF FFFF
Size	**Decimal real number (positive)**		**Decimal real number (negative)**	
D (Double word)	+1.175495E-38 ... +3.402823E+38		-1.175495E-38 ... -3.402823E+38	

Format identifier of constants

In many operations, your program can use constants in the byte, word or double word format. Format identifiers specify how a constant value is to be displayed (in the binary, decimal, hexadecimal or ASCII format).

Constants in the program are considered as decimal numbers if a format identifier is not specified:

2# for dual numbers
16# for hexadecimal numbers

Examples of binary constants

Example	Numerical basis	Separator	Constant
2#1101	2	#	1101

Examples of hexadecimal constants:

Example	Numerical basis	Separator	Constant
16#3FB3	16	#	3FB3

Note

Underscore characters can increase the readability in imported ASCII files.

Example: 16#A_B_C_D

13.3.2.6 Data blocks

Data block types

A distinction should be made between three types of data blocks (see also "Addressing range of the target system (Page 1201)"):

- Data blocks of the user interface

  These are used to communicate from the user program with the individual control components and are created by the system. The user accesses the interface using read and write access operations.

  Note

  According to the notation type and addressing, although they are data blocks, these DBs do not make a distinction between initial and actual values and they are also not part of the PLC user project. The information provided in the section "Properties of data blocks" does not apply to these DBs.

- Special data blocks

  They are used for special tasks (e.g. tool manager, service planner) and are available pre-configured in the Programming Tool PLC828. If the particular functionality is to be used, the corresponding DBs must be incorporated in the user program.

- User data blocks

  Users define there structure and incorporate them in the user program.

Data block properties

A data block is a block for data (initial values, actual values) and comments with the following properties:

- The data are saved in precisely the same sequence as specified by the user. This means that the inner structure of the data block is defined and if several data blocks are created with the same inner structure (i.e. the same type), then a certain data is always located at a specific location. This location is called offset and is the relative length in bytes from the beginning of the data block (DB) up to the actual piece of data.
- Initial values can be assigned to the data. When loaded into the CPU for the first time, the actual values of the DB are initialized with these initial values. The initial values are also stored in the CPU.
- The actual values of the data can also be read online from the control, changed and also saved with the project.

The data block structures are, just like the POUs, part of the project. They are compiled, saved, imported or exported with the project. Further, they are loaded with the project into or out of the target system. Data blocks only contain actual values. These can be loaded into or out of the target system independent of the project. It is absolutely necessary that the structure of the data block to be loaded into the target system matches the project opened in the Programming Tool PLC828. Data blocks are listed like POUs in their own symbol table.

If you change the CPU type for your project, then the existing data blocks are not lost. However, if you select a CPU in which the data blocks are not available, then you must observe that the variables from the data block are now displayed in their absolute address (e.g. DB9000.DBB0). Only when the target system is started is a check made as to whether this address is valid and the program is then possibly not executed.

You can rename your data blocks and change their properties, by clicking with the right mouse button in the operation tree on the corresponding data block. Select "Properties". The "Properties data block" dialog box opens. Here, you can change the name, block number and data class as well as add an author and comments.

Note

If you just want to rename your data block, click the right mouse button in the instruction tree on the object you want to rename and chose "Rename".

You can also easily edit your data block in another program (e.g. Microsoft Excel).

Editing data blocks in the Programming Tool PLC828

To edit data blocks in the Programming Tool PLC828:

- In the navigation bar, click on the button "Data block".
- Select the "View" > "Data block" menu command.
- In the operation tree, double click on the button "Data block" and then on the data block to be edited:

Assigning addresses and initial values in the data block

You define the structure of the DB (the sequence of the individual DB variables) in the declaration view. Initial values can be assigned to the variables of the data block that are saved with the data class or the project and with which the actual values of the DBs are initialized when loaded into the CPU for the first time.

When you define a variable, you assign its name and data type. The default initial value is set to zero/OFF, however, this can also be changed. You can optionally insert comments. The Programming Tool PLC828 automatically assigns the address. Each address is aligned by its size which means that gaps can occur.

Example:

	Address	Name	Data Type	Format	Initial Value	
1	0.0	dwDate1	DWORD	Hexadecimal	16#00000000	K1
2	4.0	wDate2	WORD	Unsigned	0	K2
3	6.0	bDate3	BYTE	Unsigned	0	K3
4	7.0	boDate4	BOOL	Bit	OFF	K4
5	7.1	boDate5	BOOL	Bit	OFF	K5
6	8.0	rDate6	REAL	Floating Point	0.5	K6
7	12.0	iDate7	INT	Signed	+0	K7

DB_9000 DB_9001 **DB_9002** DB_9003 TM_CTS TM_VTS TM_ACK

The maximum size of a data block is limited (512 bytes), if it is exceeded the Programming Tool PLC828 marks the excess variable addresses using a red wavy line. The Programming Tool PLC828 returns an error when compiling the project.

Assigning actual values

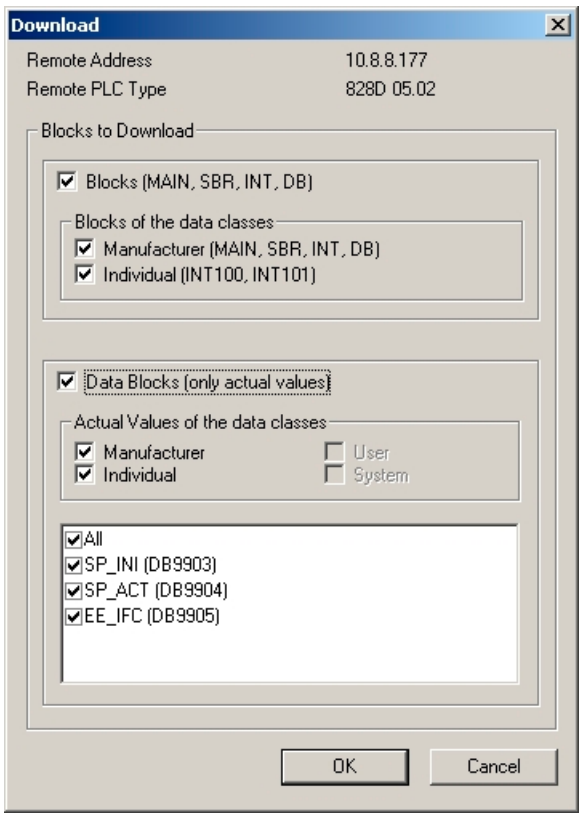
Actual values are saved with their data class (Manufacturer, Individual or User). They are only displayed and can be changed in the data view, while on the other hand, the structure of the variables (name, data type, initial value and comment) is write-protected in the data view.

After loading the actual values of the data block (not the complete project), these values become effective in the target system.

Example:

	Address	Name	Data Type	Format	Initial Value	Actual Value	
1	0.0	dwDate1	DWORD	Hexadecimal	16#00000000	16#01234567	K1
2	4.0	wDate2	WORD	Unsigned	0	0	K2
3	6.0	bDate3	BYTE	Unsigned	0	0	K3
4	7.0	boDate4	BOOL	Bit	OFF	OFF	K4
5	7.1	boDate5	BOOL	Bit	OFF	OFF	K5
6	8.0	rDate6	REAL	Floating Point	0.5	0.0	K6
7	12.0	iDate7	INT	Signed	+0	+0	K7

DB_9000 DB_9001 **DB_9002** DB_9003 TM_CTS TM_VTS TM_ACK MP_INI MP_ACT



The structure of the data block in the project must match the structure of the data block in the target system. If you have modified the structure of the data block, or re-created the DB, then the complete project has to be loaded again.

You can access the data view in the following ways:

- Click in the standard function bar on the symbol "Display data block values".

or

- Select the menu command "View" > "Display data block values".

Accepting initial values

If you accept the initial values, then all of the actual values of a data block are overwritten. If you hadn't selected an initial value for a variable, then the Programming Tool PLC828 sets it to zero/OFF. Other data blocks are not changed in the process.

In order to reset all actual values in the actual data block to the initial values:

- In the table, click on the right mouse button and in the context menu select "Accept initial values".

or

- Select the menu command "Edit" > "Accept initial values".

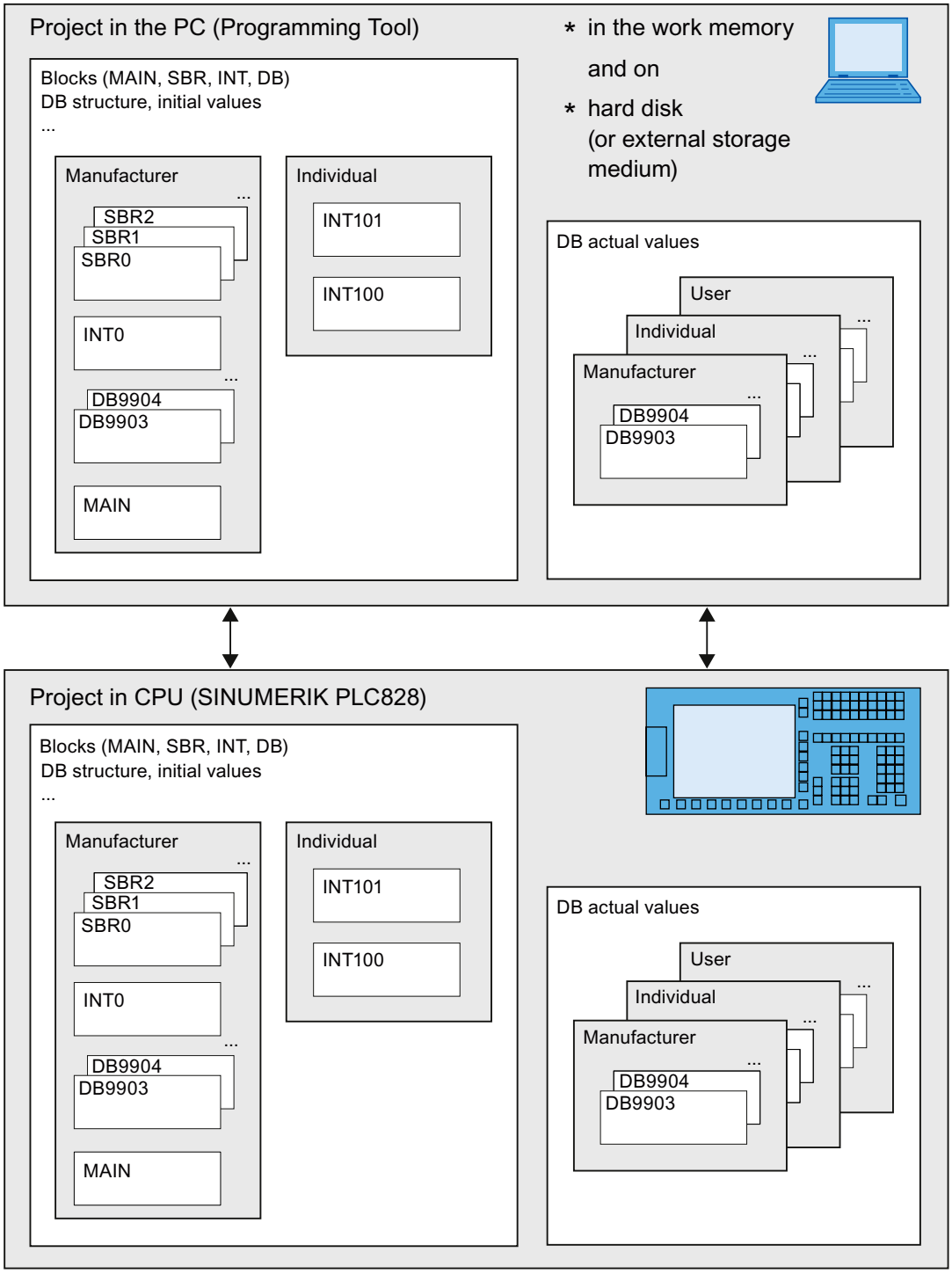
Loading and saving the data blocks

Save involves writing the DB (depending on the selection, structure + initial values + additional blocks and tables or actual values or both together) from the PC work memory to the hard disk or to another external storage medium. After the project has been opened, these values are again available.

Load into CPU involves writing the DB (depending on the selection, structure + initial values + additional blocks and tables or actual values or both together) from the PC work memory into the PLC828.

Load from CPU is the inverse operation.

Load operations are selected from the data class perspective.


Project in the PC (Programming Tool)
* in the work memory
and on
* hard disk
(or external storage medium)
Blocks (MAIN, SBR, INT, DB)
DB structure, initial values
...
Manufacturer
...
SBR2
SBR1
SBR0
INT0
...
DB9904
DB9903
MAIN
Individual
INT101
INT100
DB actual values
User
Individual
Manufacturer
DB9904
DB9903
Project in CPU (SINUMERIK PLC828)
Blocks (MAIN, SBR, INT, DB)
DB structure, initial values
...
Manufacturer
...
SBR2
SBR1
SBR0
INT0
...
DB9904
DB9903
MAIN
Individual
INT101
INT100
DB actual values
User
Individual
Manufacturer
DB9904
DB9903

Direct addressing for data blocks

- Absolute addressing

  The direct, absolute address of the variables in their data block number comprises the absolute input from the number of the data block (e.g. DB9000) a period and the address of the variables.

  The selection is made using the menu command "View" > "DB address representation (Ctrl+B)":

  Example:

  DB9000.DBB0

- Symbolic addressing

  The variables of the data blocks can be assigned names in the symbol table. When using the variable it is sufficient to specify this name from the symbol table.

  If no name is assigned to the DB variable in the symbol table, then the symbolic address comprises the DB name (e.g. myDB_9000) and the name of the variables in the data block (e.g. Vcorrection1).

Address	Name of the variables in the data block	Entry in the symbol table	Absolute representation	Symbolic representation
DB9000.DBX0.0	Vcorrection1	FeedCorrAx1	DB9000.DBX0.0	FeedCorrAx1
DB9000.DBX0.0	Vcorrection1		DB9000.DBX0.0	myDB_9000. Vcorrection1

Figure 13-3 Symbolic address representation

The menu command "View" > "Symbolic addressing" (Ctrl+Y)" is used to switch over between absolute and symbolic representation.

Indirect addressing for data blocks

You may be able to simplify the programming if data blocks with the same structure are used in your program. You can indirectly address the data blocks using the accumulators AC0 to AC3. The accumulators are used to tell the program which data block is to be handled. The value in the AC is then treated as index.

For example, for axis DBs, the program text can be somewhat reduced by not having to write a dedicated program for each axis, but instead access the appropriate axis via the various data blocks and the index (AC). The value in AC is treated as an index. This is the reason that it starts at 0 for the first axis. Indirect addressing is only possible using ACs. It is not possible to display the actual data value in the program status of the Programming Tool PLC828. The absolute address cannot be determined. V addresses cannot be used for indirect addressing.

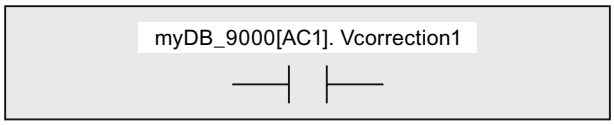


Figure 13-4 Indirect addressing

Absolute input	Symbolic input
DB3800[AC1].DBX2.1	ToAxis[AC1].ControlEnable
DB9000[AC0].DBW0	Prototyp1[AC0].MyWord1

It is not permissible to use constants for indexing:

DB3800[1].DBX2.1	ToAxis[5].ControlEnable

Indirect addressing using V addresses is also not permitted:

V3800[5]0002.1	V380[5]0002.1
V3800[AC0]0002.1	V38[AC0]0002.1

Using Cut, Copy and Paste in the data block editor

You can save data (rows, columns and cells) of your data block in the Microsoft clipboard to edit them in a different program. You can proceed as follows:

- In the context menu (right mouse button) using the commands "Cut"/"Copy" and "Paste".
- Using the keyboard by pressing the shortcut keys <Ctrl+X>/<Ctrl+C> and <Ctrl+V>.
- In the main menu using the "Edit" > "Cut"/"Copy" and "Edit" >"Paste".

Cut:	Selected data is copied into the clipboard and is deleted
Copy:	Selected data is copied into the clipboard and is not deleted
Paste:	If data is in the clipboard, then this data is pasted.
Delete:	Selected data block/section is deleted, it is not saved in the clipboard.

You can easily edit your data block in this way, e.g. in Microsoft Excel.

	A	B	C	D	E	F	G
1	DBB0	mybyte1	BYTE	Unsigned	0	0	Comment for DI
2	DBB1	mybyte2	BYTE	Unsigned	0	0	Comment for DI
3	DBW2	myword1	WORD	Unsigned	0	0	Comment for DI
4	DBW4	myint1	INT	Signed	0	0	Comment for DI
5	DBX6.0	mybit1	BOOL	Bit	OFF	OFF	Comment for DI
6	DBX6.1	mybit2	BOOL	Bit	OFF	OFF	Comment for DI
7	DBX6.2	mybit3	BOOL	Bit	OFF	OFF	Comment for DI
8	DBX6.3	mybit4	BOOL	Bit	OFF	OFF	Comment for DI
9	DBX6.4	mybit5	BOOL	Bit	OFF	OFF	Comment for DI
10	DBX6.5	mybit6	BOOL	Bit	OFF	OFF	Comment for DI
11	DBX6.6	mybit7	BOOL	Bit	OFF	OFF	Comment for DI
12	DBX6.7	mybit8	BOOL	Bit	OFF	OFF	Comment for DI
13	DBD8	mydword1	DWORD	Unsigned	0	0	Comment for DI
14	DBD12	mydint1	DINT	Signed	0	0	Comment for DI
15	DBD16	myreal1	REAL	Floating Point	0.0	0.0	Comment for DI

Using special data blocks

The Programming Tool PLC828 offers you the possibility of using special data blocks for tool change, maintenance planning and the device manager. You will find these in the operation tree under "Libraries" > "Special data blocks".

These data blocks have a fixed structure. They can be used with a double click or using Copy and Paste (in the operation tree).

The following special data blocks are available:

TM_CTS	(DB9900)	Constant transfer step table for tool change
TM_VTS	(DB9901)	Variable transfer step table for tool change
TM_ACK	(DB9902)	Acknowledgment step table for tool change
SP_INI	(DB9903)	Start data for maintenance planner
SP_ACT	(DB9904)	Actual data for maintenance planner
EE_IFC	(DB9905)	Interface for device manager

Resolving errors

The Programming Tool PLC828 marks errors made when data is being entered (e.g. in the LAD Editor or the symbol table). For example, an illegal syntax or the use of invalid values can result in input errors. You must correct all errors that have occurred before you can compile error-free and load the program into the CPU.

Every data block is involved in the compilation run of the project.

Compilation is started:

- Using the menu command "Target system" > "Compile"

or

- by clicking on the symbol.

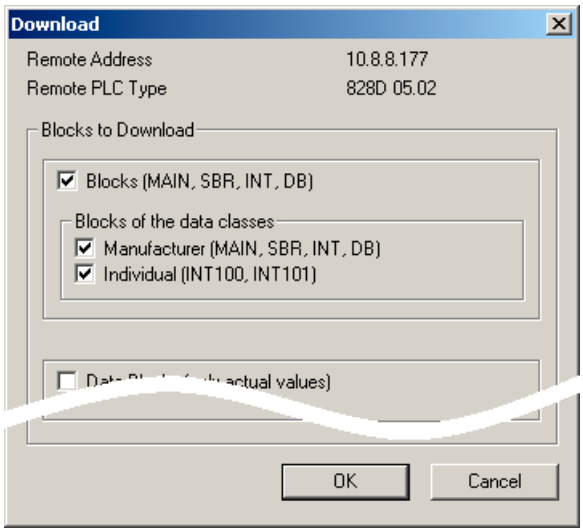
If any errors occur during compilation, then they are displayed in the output window. Position the cursor to an error message in the output window and double click on it so that you see the line in the data block with the error.

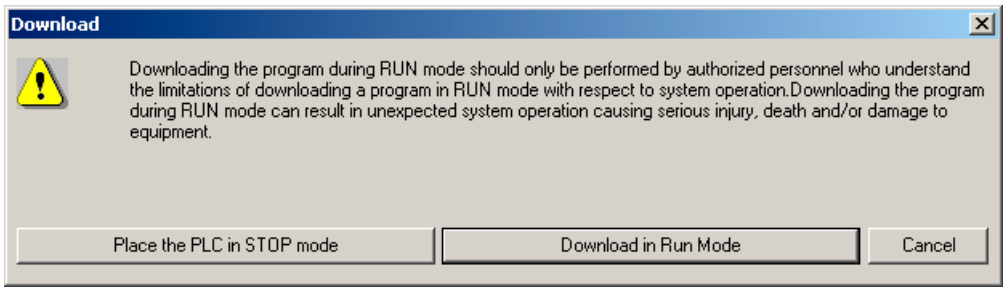
Loading the data block into the target system

Here, reference is made to Chapter "Data classes (Page 1225)" in which loading and saving (not only data blocks) is explained in detail.

- After making structural changes to a data block, this must be loaded into the target system with the PLC in the stop state. The changes only become effective when subsequently going into the RUN operating state. If the target system identifies that a data block is new or has been changed, then it sets the initial values for this data block as the first actual values.

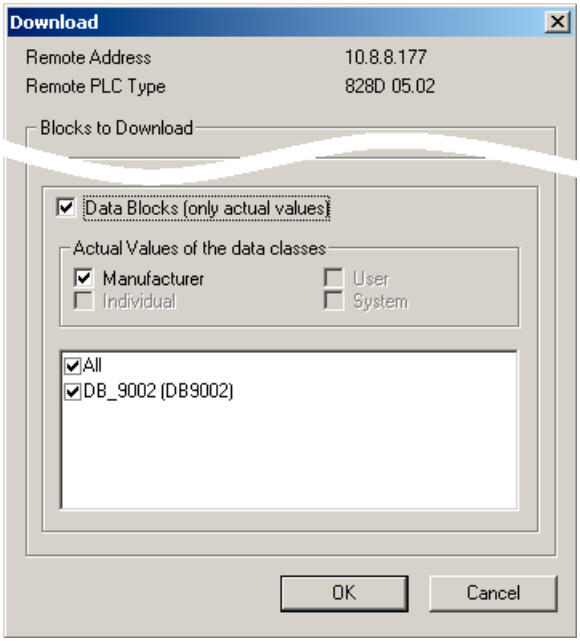
  Under all circumstances, you should activate the three check boxes of the data classes. Why? If the modified DB is also called from the data class "Individual", then program errors occur as a result of the fact that it is deselected: Namely then, INT100 or INT101 access the "old" structure of the DB.





- After making changes to the actual values of a data block, these must then be subsequently loaded into the target system. (The structure of the data block in the target system must match the structure of the data block in the project). There are two ways of loading the actual values into the CPU:

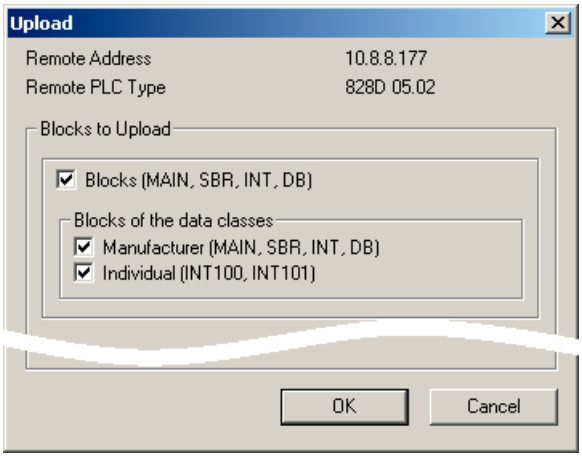
  – By clicking on the "Write all" symbol.
  – By loading the DB in the RUN operating state:



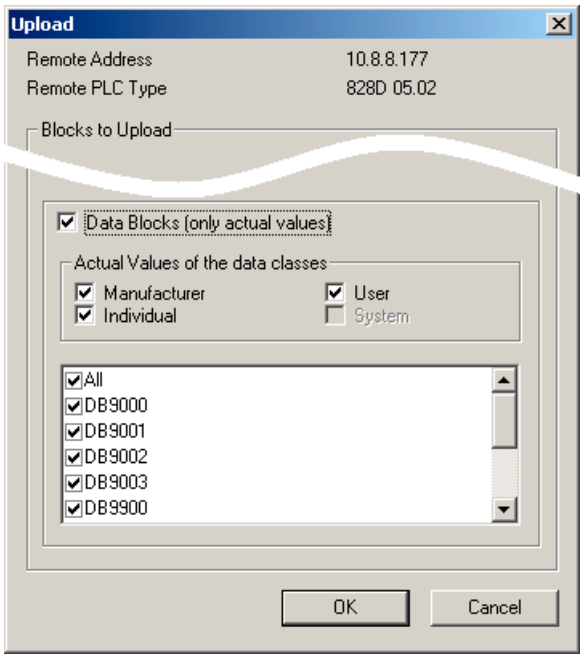
Loading a data block from the target system

Here, reference is made to Chapter "Data classes (Page 1225)" in which loading and saving (not only data blocks) is explained in detail.

- You must first open a project in the Programming Tool PLC828 before you can load the program block (project) and therefore its structure from the target system. As the structure and initial values of the data blocks are permanently assigned the "Manufacturer" data class, then only the check box of this data class has to be activated.



- If you only want to load the actual values of a data block from the target system, then first open the corresponding project in the Programming Tool PLC828 or load it from the target system. You can now load the actual values. You cannot load the data block if the structure of the data block in the target system does not match that of the data block of the project that has been opened, or if the project that has been opened does not have a data block.



13.3.2.7 Special bit memories and their functions

Special bit memories SMB0 (SM0.0 ... SM0.6) contain seven bits that are updated by the PLC firmware at the end of each cycle. You can implement various functions in your program with these bits.

Special bit memories (write protected)	Description
SM0.0	This bit is always switched on.
SM0.1	This bit is switched on in the first cycle. It is used, for example, to call an initialization subprogram.
SM0.2	This bit is switched on for the length of a cycle, if retentive data has been lost. It can be used either as an error bit memory or as a mechanism for calling up special startup sequences.
SM0.3	This bit is switched on for the length of a cycle if the RUN operating mode has been set after switching on (power on). This permits a warm-up time for the system before starting operation.
SM0.4	This bit ensures a cycle that is switched-on for 30 seconds and switched-off for 30 seconds. And more precisely for a cycle time of 1 minute. This way you have an easy to program delay or a cycle time of 1 minute.
SM0.5	This bit ensures a cycle that is switched-on for 0.5 seconds and switched-off for 0.5 seconds. And more precisely for a cycle time of 1 second. This means that you have an easy way of programming a delay time or a cycle time of 1 second.
SM0.6	This bit represents a clock cycle. It is switched-on for one cycle and switched-off for the next cycle. So you can use this bit as a cycle counter input.
SM0.7	Reserved

13.3.3 Operation set

The PLC of SINUMERIK 828D provides the following operation groups:

- Bit logic operations
- Fixed-point arithmetic
- Interrupt operations
- Floating-point arithmetic
- Program control operations
- Shift/rotate operations
- Transfer operations
- Conversion operations
- Comparison operations
- Logical operations
- Counters
- Timers
- Subprograms

You can take details from the online help of the Programming Tool PLC828 and the S7-200 System Manual.

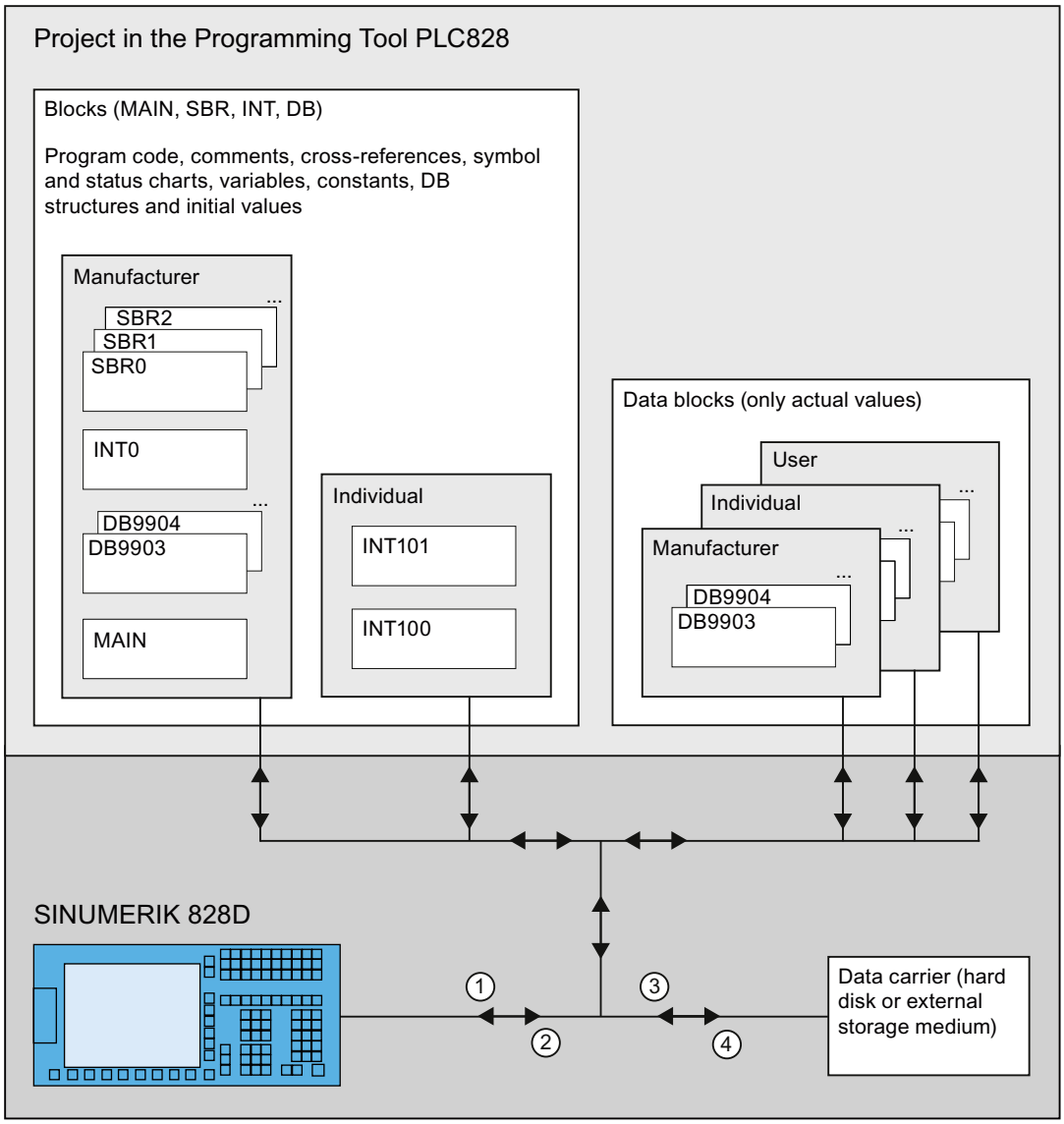
13.3.4 Data classes

13.3.4.1 Defining data classes

Overview

Data classes are user-related organizational units for programs and data blocks as project subcontainer.

They are special in so much that they mutually demarcate the contents: The data and programs assigned to them can be handled as group - i.e. as data class. This applies when loading into the control and from the control as well as for data backup using export and import.



① Load into CPU
② Load from CPU
③ Open Project
④ Save project (under)

Figure 13-5 Project structure in the Programming Tool PLC828 and the structure of the data class transport paths

From the user's perspective, there are three data classes:

- Manufacturer:
  - POUs (MAIN, all subprograms and the interrupt program INT0);
  - The data blocks: internal structure (type information) plus initial values;
  - The data blocks: Actual values, if assigned by the user;
- Individual:
  - the interrupt programs INT100 and INT101
  - The data blocks: Actual values, if assigned by the user;
- User:
  - The data blocks: Actual values, if assigned by the user;

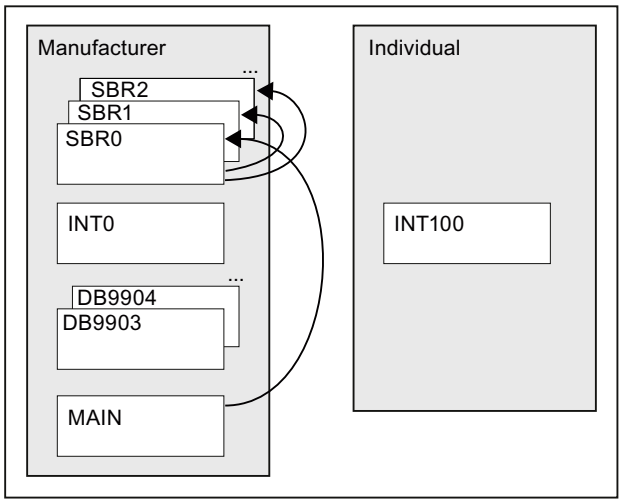
All of the existing data classes are always automatically selected in the dialog box for loading. This means that all of the data and blocks belonging to the project are always loaded, if they have not been deselected by the user.

Example

With a standard (series) machine, where the machinery OEM also commissioned the PLC user program, a PLC input (IO module) fails. The input is rewired to a free input. The INT100 interrupt program can be used so that the service/erection technician doesn't have to change the PLC user program of the machine manufacturer due to the rewiring, which is a complex and tedious task: This runs in front of the main program (MAIN) and writes the rewired input to the original input in the image. INT100 is assigned to data class "Individual" and is loaded into the CPU with this data class.

The project block of the machinery construction OEM has the data class "Manufacturer" and is loaded into the CPU with this data class.

If the machine manufacturer updates his PLC user program, the correction program is not influenced. The corrections remain independent of this and are still effective.



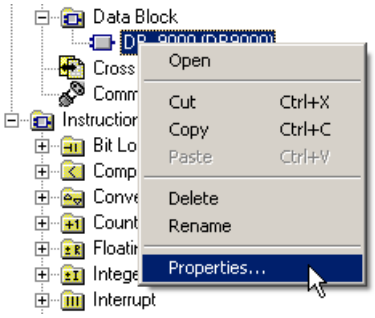
The interrupt program INT100 can be independently loaded in its data class and the correction function described above executed.

13.3.4.2 Assigning a block to a data class

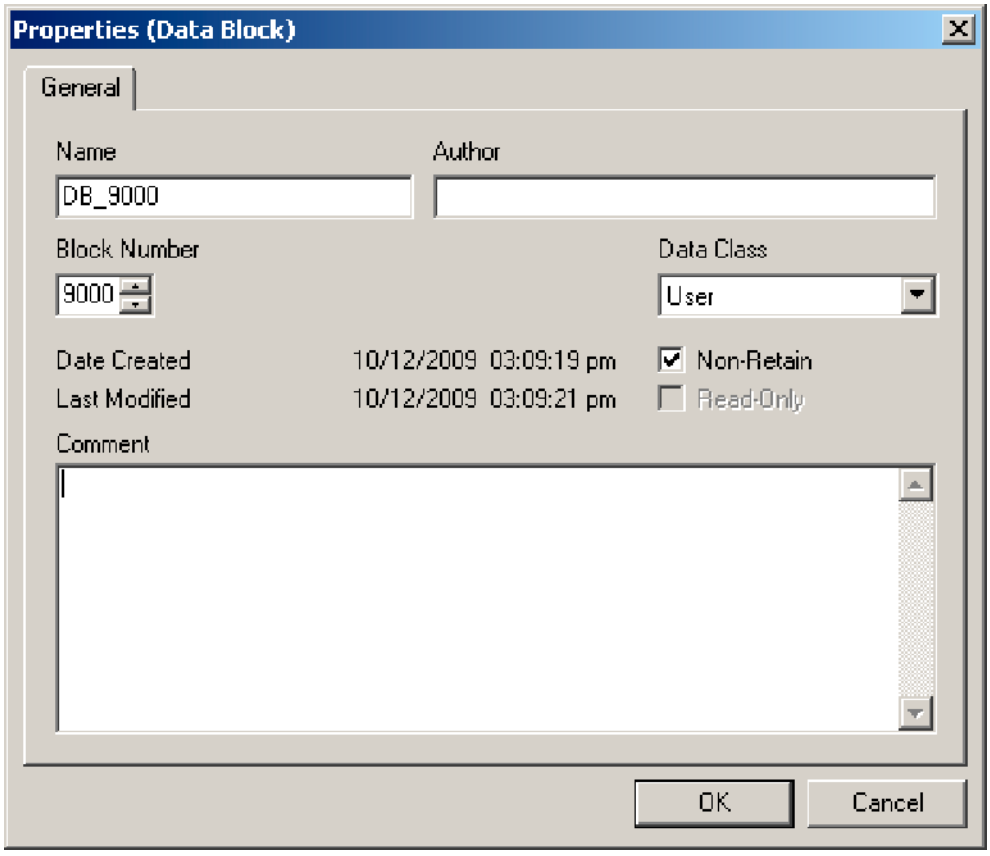
A data block is assigned to a data class in its property dialog box.

Procedure

1. In the operation tree, right click on the corresponding block and select "Properties":



2. Assign the block to one of the three possible data classes:



Here, for DB9000, in addition to the data class "User", the "Non-Retain" property was also selected. Data blocks with this attribute are reset to the initial values after each power off and power on.

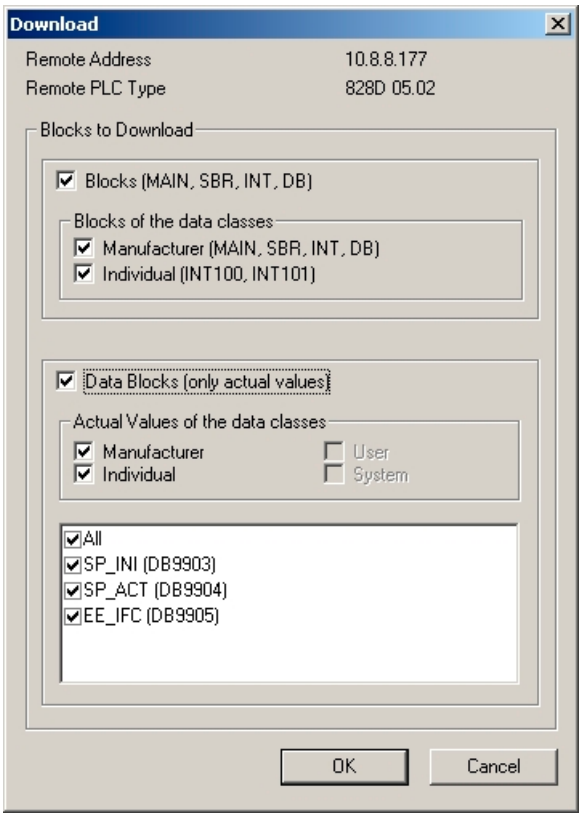
See also

Defining data classes (Page 1225)

13.3.4.3 Load data class(es) into the CPU

Procedure

1. In the window "Load into CPU", select the data class(es) whose blocks are to be loaded:



2. Select the option "Blocks (MAIN, SUBR, INT, DB)", if program or data block changes are to be loaded into the target system.
3. Select the option "Data blocks (only actual values)" if actual values of one or several data blocks are to be loaded into the target system.

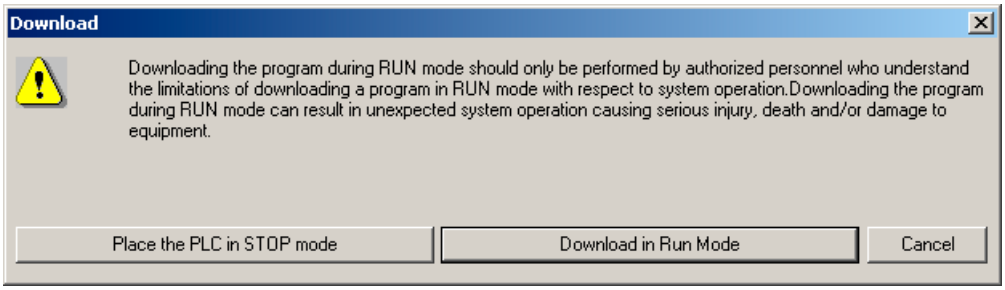
4. Select one of the following options in the message window "Load into CPU":
   – "Bring target system into the stop condition"

     → If structurally modified programs or initial values of data blocks are to be loaded into the target system.

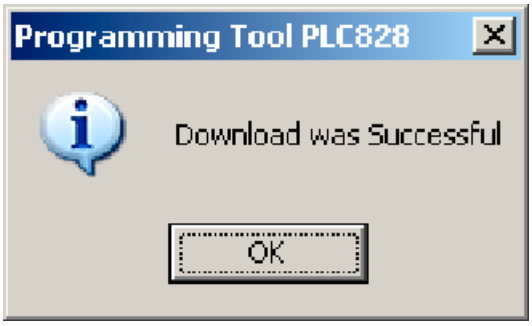
     or

   – "Load in the RUN operating state"

     → If neither program nor data block structures have changed.



The following message must be output after loading:

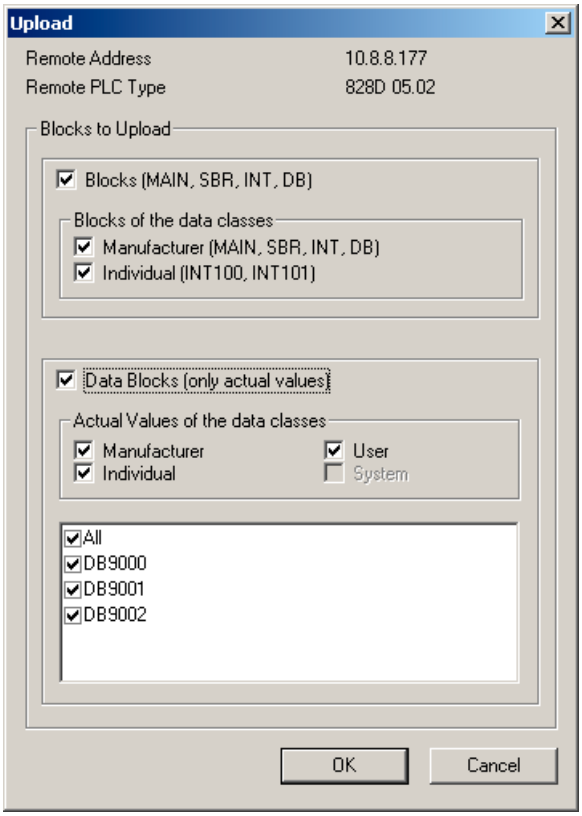


5. Confirm this message.
6. After "Loading in stop condition" – if required – switch the control back to RUN.

13.3.4.4 Load data class(es) from CPU

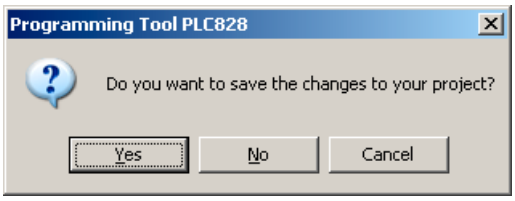
Procedure

1. In the window "Load from CPU", select the data class(es) whose blocks are to be loaded:



2. Select the option "Blocks (MAIN, SUBR, INT, DB)", if program or data block changes are to be loaded into the target system.
3. Select the option "Data blocks (only actual values)" if actual values of one or several data blocks are to be loaded into the target system.

4. If required, save the opened project in the Programming Tool PLC828, it is then overwritten when "Load from CPU":



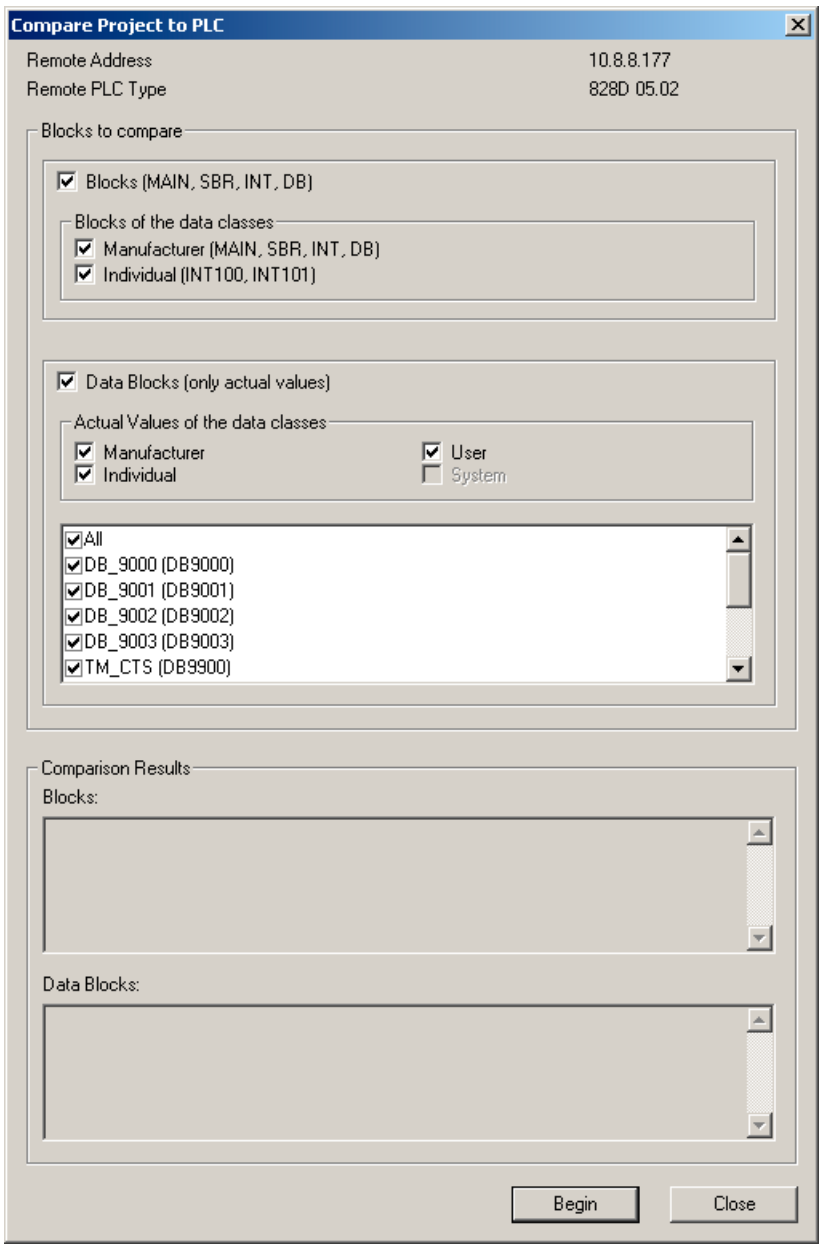
   A new project with the name of the project located in the target system is created.

5. After a warning that the existing project will be possibly changed as a result of "Load from CPU", the requested data are loaded from the target system (also refer to the diagram "Project structure in the Programming Tool PLC828 and the structure of the data classes transport paths" in Defining data classes (Page 1225)").

13.3.4.5 Comparison between online and offline projects

The data classes, whose blocks are to be compared, can be selected in the dialog box "Compare..."(menu "Target system" > "Compare...").

If data classes only exist in the offline project or only in the CPU (online project), these are correspondingly marked. The differences that exist between the offline and online existing program blocks (SBRs, INTs) or data blocks (DBs) are shown as a result of the comparison:



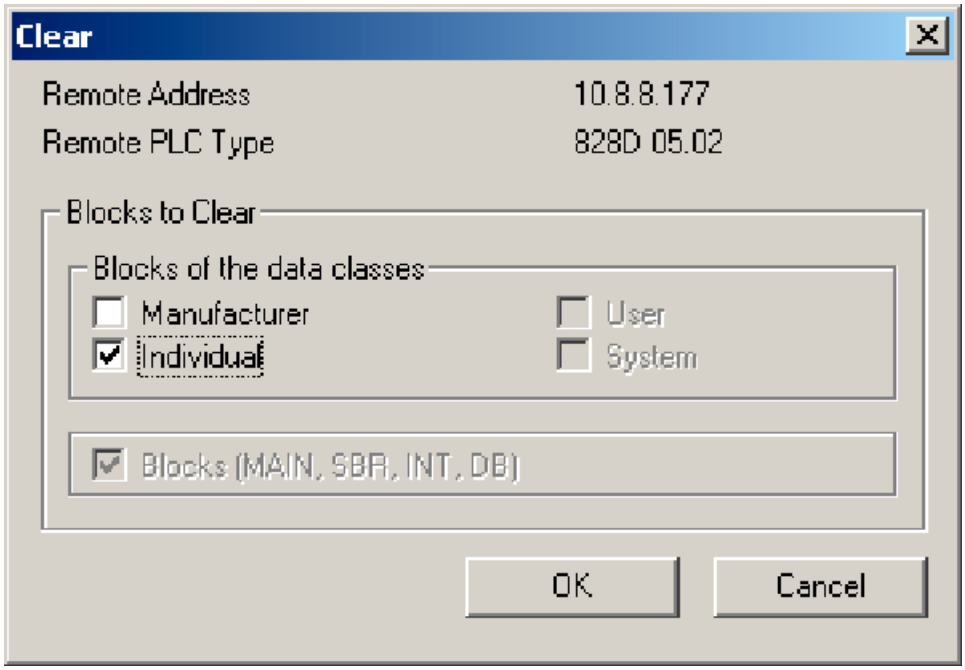
Example:

The differences between the actual values saved (offline) in the project (e.g. +22) and the actual values available in the target system (online) (e.g. +11) are displayed in the lower field "data blocks".

13.3.4.6 Delete in the target system

Delete is only permitted in the "STOP" operating state.

Deleting "Manufacturer" results in a start error at the transition into the "RUN" operating state. The target system then returns into the safe "STOP" operating state.



See also

Defining data classes (Page 1225)

13.3.5 Rewire addresses

Function

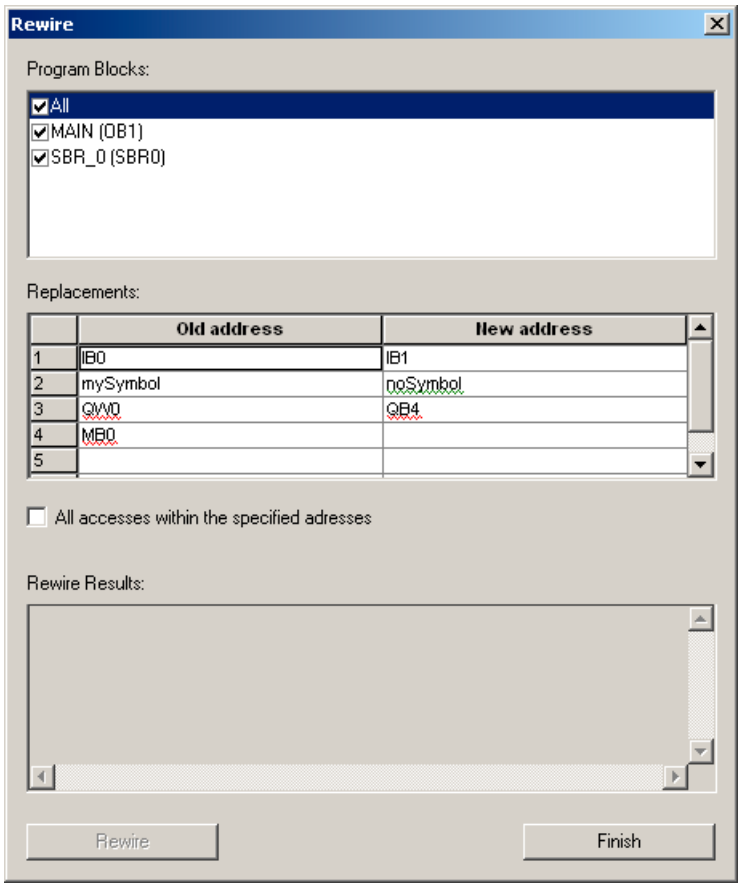
Addresses in the PLC user program can be centrally changed - e.g. IW0 to IW8 - using the "rewire" function. This means that user programs can be quickly adapted to the modified I/O expansion stage.

Example:

The customer writes a PLC user program for a series of machines. Due to the different machine expansion stage, the I/O expansion stage of several machines differs, which is the reason that addresses must be individually changed in the user program. Under certain circumstances, this can involve several hundred addresses. Using the "rewire" dialog box, for these machines, it is now possible to enter a list of the addresses to be changed, e.g. inputs and outputs. This is executed using the "rewire" function, and the addresses are changed in the user program.

Procedure

Use the dialog box "rewire", to rewire addresses:



Please proceed as follows:

1. Open the dialog box "Rewire" in the LAD editor using the context menu ("Rewire …") or using the menu bar ("Edit > Rewire …").
2. In the list "Program blocks" (list of all of the POUs available in the project), select the POUs in which the rewire operation is to be executed.

3. Enter the old and the new addresses for rewiring in the "Replacements" list.

   Permitted addresses include:

   – Inputs
   – Outputs
   – Bit memory
   – Special bit memory
   – Variable memory / data blocks
   – Timers
   – Counters

   Editing the list of addresses

   The following functions are supported using the context menu (right mouse button):

   – Cut (Ctrl+X)
   – Copy (Ctrl+C)
   – Paste (Ctrl+V)
   – Select all (Ctrl+A)
   – Paste row (Ctrl+I)
   – Delete selection

   This means that it is possible to copy the list or parts of the list from other or into other applications, e.g. Microsoft Excel.

   "Old address"

   In this column, enter the name or the address which you wish to rewire.

   "New address"

   In this column, enter the new name or the new address. Please ensure that the type of the new address corresponds to that of the old address, e.g. old address IW0 and new address IW4, not IB4, or old address DB9000.DBB0 and new address MB0, not MW0.

   Checking the validity of addresses

   If the name of the address (symbol) does not exist in the open project, then this is marked with a green wavy line.

   If the type of the old address does not match that of the new address, or if only one address was entered (old or new address), then this address is marked using a red wavy line.

4. Select or deselect the option "All accesses within the specified addresses".

   If the option is enabled, then the addresses ranges (BYTE, WORD, DWORD) are rewired.

   Example:

   You specify IW0 and IW4 as address ranges. The, addresses I0.0 ... I1.7 are rewired to addresses I4.0 ... I5.7. Addresses from the rewired range (e.g. I0.1) can then no longer be individually entered into the table.

5. Click on the button "Rewire" to start the function.

Note

If you wish to exit the dialog box, without activating the function "Rewire", then use the "Exit" button.

After executing the rewire function, the results are displayed in the list "Results of rewiring". The list contains the address list with the columns "Old address" and "New address". These list the individual blocks and the number of wiring operations that were carried out in each block. Using the context menu (right mouse button), the results can be copied into other applications, e.g. Microsoft Word.

Note

The following must be taken into account when rewiring:

- Name or number of a POU cannot be changed using the "Rewire" function. For this purpose, in the operation tree in the POU context menu (click with the right mouse button, e.g. on SBR_0) using the functions "Rename" or "Properties ...".
- Timers can only be rewired to remain timers (e.g. old address T0, new address T16) and counters can only be rewired to remain counters (e.g. old address C0, new address C1).

13.4 Test and diagnostic functions

13.4.1 Program status

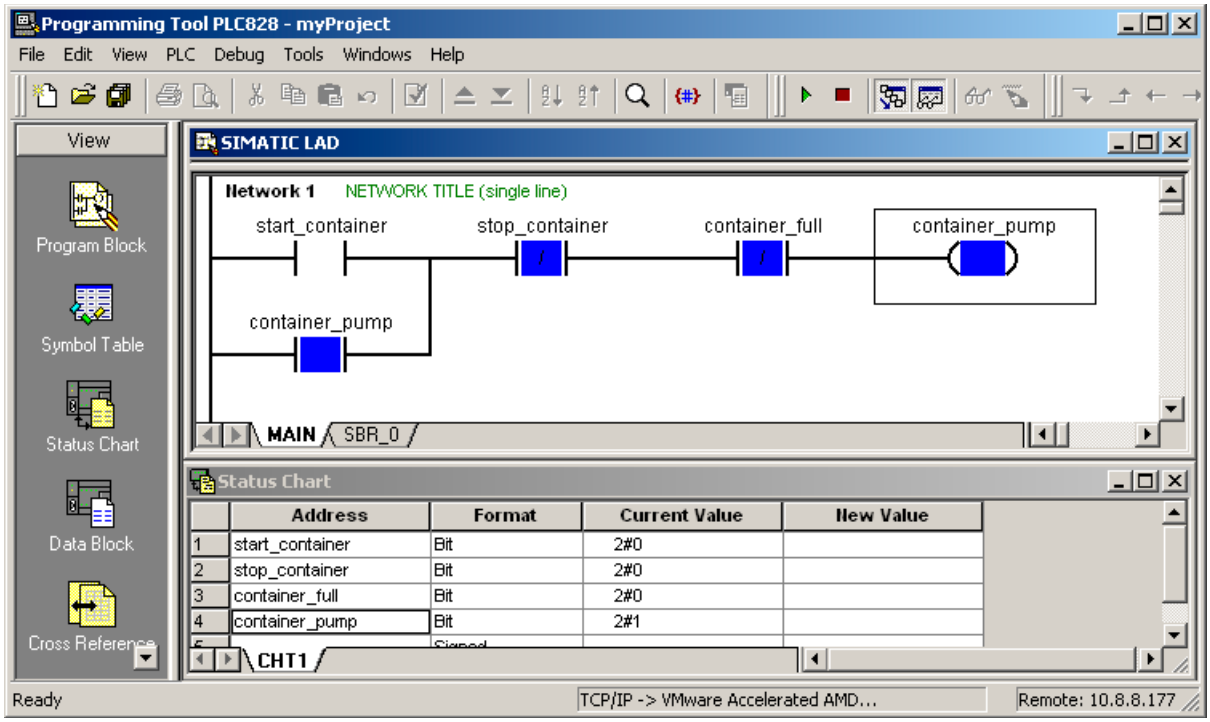
Once you have established communication between your programming device, on which the Programming Tool PLC828 is installed, and a target system and have loaded a program into the target system, you can work with the diagnostic functions of the Programming Tool PLC828 and test new programs as well as monitor programs that are already being processed.

These help topics are subsequently described.

13.4.1.1 Status definition

"Status" refers to the display of the actual values of addresses while executing the program in the target system. You can display status information in a status chart or by switching-on the program status in the program editor.

An example for status information in the status chart and in the program editor of the Programming Tool PLC828 is shown in the following diagram:



Note

Please note that unnecessary project components (e.g. the operation tree and the output window) have been omitted, in order to provide more space to display the required components (LAD program editor, status chart, function bar to test and symbol for the status chart in the navigation bar). Using the menu "View", you can set-up the environment in the Programming Tool PLC828 corresponding to your particular tasks, i.e. you can display just the project components that you require.

13.4.1.2 Preconditions of the status update

Before you can update the status to monitor and test your program, you must execute the following tasks:

- Your program must be able to be compiled error-free.
- You must have set-up the communication between the Programming Tool PLC828 and the target system.
- Your program must have been loaded error-free into the target system.
- After you have loaded the program into the target system, then you should again bring this into the RUN operating state. Otherwise, the address status is displayed, however, the target system cannot execute the program so that you are not shown the logic operations that you expect.

13.4.1.3 Influence of the operating state on the target system

The type of monitoring and test functions that you execute depends on the operating state of your target system.

Even if your program is not executed in the STOP operating state, the operating system of the target system still monitors the target system (status of RAM and I/O, and transfers the data status to the Programming Tool PLC828. If the target system is in the STOP operating state, then you can execute the following functions:

- You can display the actual values of the addresses in the table status or in the program status. (This is the same as the function "Single read", as the program is not executed.)
- You can write values in the table status.
- You can execute a certain number of cycles and display the effect in a status chart and/or in the program status.

If the target system is in the RUN operating state, you cannot execute the functions "First cycle" or "Several cycles". You can write values into a status chart, you can also execute the following functions (not in the STOP operating state):

- In the table status, you can carry out the continuous updates. (If you wish to only execute an update, you must switch-off the table status so that you can execute the command "Single read".)
- You can execute continuous updates in the program status.

13.4.1.4 Communication and cycle

In a continuous cycle, the target system reads the inputs, it executes the program, writes to the outputs and executes system functions as well as the communication. This cycle runs with an extremely high speed of many times per second. Even if the Programming Tool PLC828 issues status requests in a fast sequence, it is important that you clearly understand that you cannot monitor each individual event that takes place in the target system. When using the program status or the table status, if you read data values from a target system program, the interrogate the data by taking samples (spot check). The update rate of the status values read from the target system depends on the communication baud rate.

13.4.1.5 Status update

Program status in LAD

If you monitor the program status in the program editor in LAD, the status is updated at the end of each cycle. If an address is processed by several operations, the intermediate values of the addresses are not displayed by the status. Only the values of the addresses at the end of each program cycle are displayed in the LAD program status.

Update status (procedure)

You can update the status in various ways:

- Open the program editor window and enable the program status (menu "Test" > "Program status") in order to view the continuous status update in the RUN operating status of the target system.

  Bear in mind that "continuous" does not mean in real time; instead, it means that the programming device quickly polls the PLC for status information and displays it on your screen, updating the display as quickly as your communications permit. Some rapidly fluctuating values may not be identified and displayed on your screen. It is also possible that the values change too quickly for you to read them. You can update the status once if you switch the target system to the STOP operating state. Even if the target system is in the STOP operating state, you can use the "Multiple cycles" command to view one or more cycles. Using the "First cycle" function, you can view a single cycle - whereby the bit memory of the first cycle is activated.

- Open the status chart window and enable the chart status to view continuous updates when the target system is in the RUN operating status.

- Disable the chart status and use the "Single read" function if you wish to update the status and you do not want to switch the target system into the STOP mode.

  If you switch the target system into the STOP operating state and enable chart status, then this also enables you to update the status. Furthermore, you can use the "Multiple cycles" and "First cycle" functions while you are viewing a status chart.

13.4.1.6 Simulating process conditions

You can simulate process conditions by writing new values to addresses. To do so, use the status chart.

13.4.1.7 Checking cross references and the elements used

If you test your program, it is possible that you wish to supplement, delete or change the parameters.

In the window "Cross-references", you can determine how the parameters are presently assigned in your program. This helps you to avoid assigning values twice.

13.4.2 Program status in the LAD program editor

13.4.2.1 Display program status

Note

If, in the STOP operating state, you have loaded a program into the target system, you must switch the target system back into the RUN operating state before you can display continuous updates of the program status!

Procedure

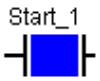
To enable the program status, proceed in one of the following ways:

- Select the menu command "Test" > "Program status".

or

- To test, click in the function bar on the "Program status" button.

The program status is displayed in the program editor.



Boolean operations (contacts, coils) are displayed as colored blocks if the address value is 1 (the bit is enabled).

The value of non-boolean addresses is displayed and updated as quickly as the communication permits it.

Note

If you enable the program status, many other functions in the programming tool are deactivated. For instance, you cannot change your program, unless you disable the program status again. Other functions, e.g. switching the display from one program editor to another, mean that the program status is automatically disabled. If you wish to display the status gain, you must reselect the command "Program status".

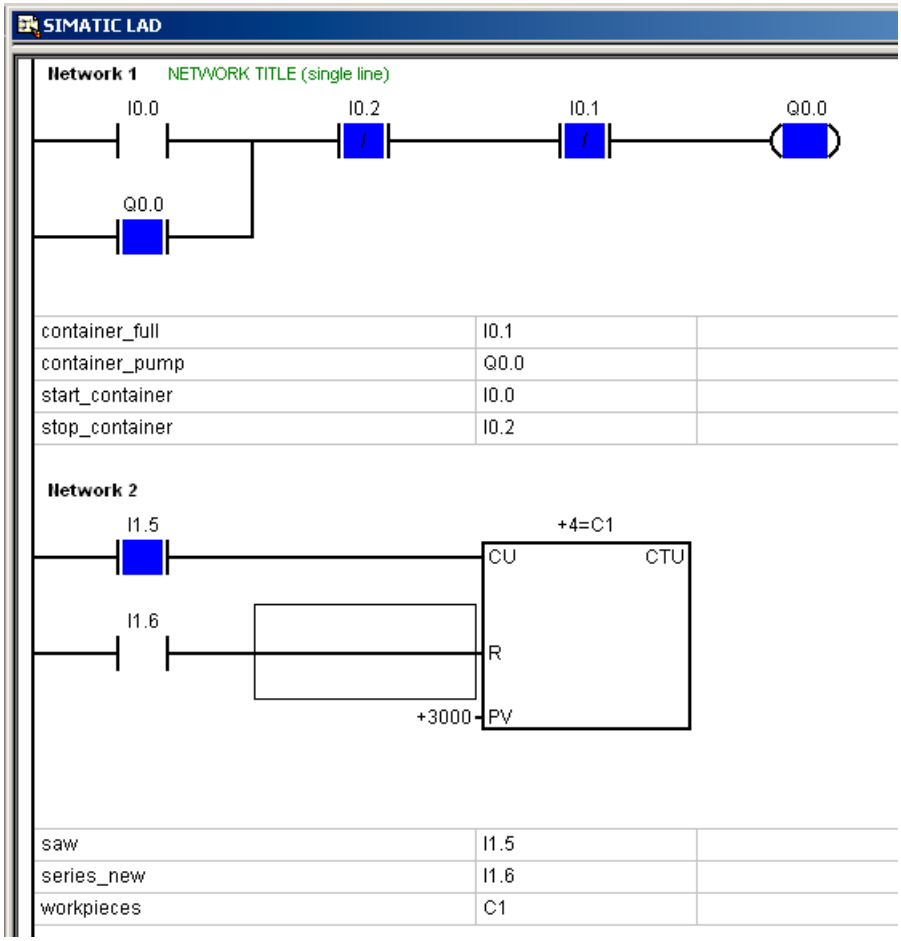
Note

If you have problems when enabling the program status, please consider the following prerequsites:

- You must have set-up the communication (so that you can load your program into the target system).
- You must have selected the correct CPU version so that you can load the program into the target system.
- Your program must be able to be compiled error-free.
- Your program must be able to be loaded error-free into the target system.
- Your target system must be in the RUN operating state in order that the status update can be continuously displayed. Otherwise, only changes at the inputs and outputs (if they are available) are displayed. As the program is not executed in the target system, changes at the inputs and outputs do not have the same effects as you would expect at the program logic in the displayed program status.
- If you display another program area, which is not executed (e.g. an interrupt program or subprogram or an area, which was skipped due to a jump operation), the status is not displayed as the code is not interrogated.

13.4.2.2 Display properties

If you display the program status in LAD, the boolean operations are displayed as colored blocks, if the value of the address is 1 (bit is enabled). The actual data value from other addresses is displayed next to the address (or instead of the address). The display is updated, if changes are read from the target system.



13.4.2.3 Restrictions

The advantage of displaying the status in the program editor is due to the fact that you obtain a graphic display of the events in your program. However, not all of the tools are available as in the status chart (e.g. the function "Single read", "Write all").

It is important that you know the restrictions of the program status if you test and monitor your program.

Signal flow

The color marking for "Signal flow" does not mean that there is always a signal flow!

In the program status in LAD, the values are only displayed at the end of the cycle. This means that it can be sometimes difficult to evaluate just what the "signal flow" display really means. Boolean contacts and coils are shown color coded in the program status in LAD corresponding to the value of the bit addresses. If the bit value is 1 (bit is enabled), then the operation is marked in color. However, this does not necessarily mean that the operation was actually executed. There are several conditions that can result in an unclear signal flow display:

- If, when evaluating the status, the target system is in the STOP operating state, contacts can be activated, however, coils and boxes are not switched-on because the program is not being executed.
- If the program contains a jump operation, then it is possible that the networks that you are investigating, do not display the expected results because the target system skipped these operations while executing the program.
- It is a similar situation if you consider a subprogram. The boolean addresses can be activated, however, the subprogram logic can only be executed if the subprogram is activated. If the subprogram was not called from the main program, then no logic of the networks was executed regardless of what the bit values of the operations display.

Status at the end of the cycle in LAD

If you display the program status in the program editor in LAD, the status is updated at the end of every cycle. Displaying your program in another programming language has no effects at all on the actual program.

13.4.2.4 Adapting the program status display

Adapting the program status display in the program editor

Procedure:

1. Select the menu command "Tools > Options".
2. Open the "Status" tab.

You can now edit the following settings:

- **Zoom factor**

  To edit the scaling.

  Shortcut key: You can use the shortcut key to quickly set the zoom factor in the program status:

  – Press the Ctrl key and the plus key in the numerical block of the keyboard to increase the display size.

  – Press the Ctrl key and the minus key in the numerical block of the keyboard to reduce the display size.

- **Field width and height**

  To edit the grid settings.

  You can increase the field width so that information can be displayed which otherwise would be cut-off. You can reduce the field height so that there is sufficient space so that your networks can be shown on the screen.

- **Signal flow**

  You can change the color, which shows that boolean addresses are activated (bit value is 1).

- **Address display**

  You can display the addresses either within or outside the operations. You can also display the status value without changing the name or the address.

Adapting the arrangement of the window in the Programming Tool PLC828

In order to create more space for the window of the program editor, or to be able to display it with another window, e.g. the status chart, the symbol table or the cross references, you can adapt the arrangement and size of all of the windows shown in the Programming Tool PLC828 as follows:

1. Select the window whose display you wish to adapt.
2. Reduce/increase the size and position the window using the mouse or using the appropriate key combinations of your keyboard.

   -or-

   Use the commands from the menus "View" and "Window" that are available for the window display.

13.4.3 Displaying the status in a status chart

13.4.3.1 Properties of a status chart

Application

After you have loaded your program into the target system, you can generate one or several status charts to monitor and test program execution.

The program is continuously executed if the target system is in the RUN operating state. You can enable the chart status so that the status values in the chart are continuously updated (not interrupted). As an alternative, using the "Single read" function, you can generate a "Snapshot" of the status values in the table without having to enable the status chart.

While you look at a status chart, you can also switch the target system into the STOP operating state and only execute the first or a certain number of cycles in which you monitor program execution.

Note

Please note that you cannot change your chart if the chart status is enabled! Disable the chart status if you wish to edit the chart.

Opening / enabling a status chart

Opening a status chart is not the same as enabling a status chart. You can open and evaluate or change a status chart: However, if you do not execute the command "Single read" (in the menu "Test" or in the function bar) or enable the chart status (in the menu "Test" or in the function bar) no status information will be displayed in the "Actual value" column.

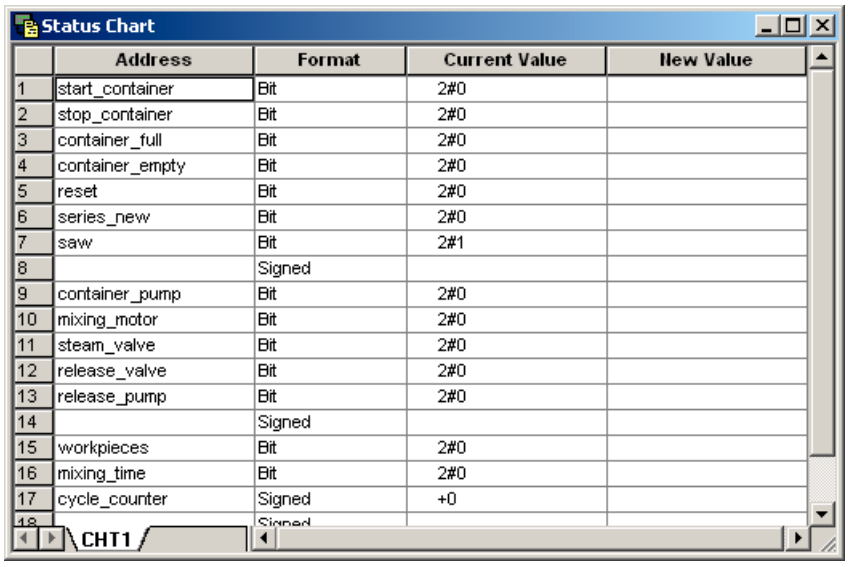


Figure 13-6 Example of a status chart

"Single read" function

If you use the function "Single read" (this is only available when the chart status is disabled) to evaluate a status chart, the actual values of the target system are accepted and displayed in the column "Actual value". However, the values are not updated while the target system executes the program.

"Chart status" function

If you enable the chart status (in the menu "Test" or in the function bar), the actual values of the target system are regularly updated. If changes are received from the target system, then the column "Actual value" is updated.

"New value" column

You can assign (write) certain values in the target system using the column (new value).

13.4.3.2 Open status chart

Open a status chart to evaluate or change the contents of the chart.

Procedure

- Click on the button "Status chart" in the navigation bar.

or

- Select the menu command "View" > "Status chart".

or

- Open the directory of the status chart in the operation tree and double click on the symbol of a chart.

- If your project includes more than one status chart, using the tab for the status charts at the lower edge of the window, you can switch over between the individual charts:

Note

If you open a status chart, then the status is still not displayed. You must enable the status chart so that the status information is updated (see "Enabling the status table (Page 1256)").

13.4.3.3 Working with several status charts

Inserting additional additional status charts

To insert additional status charts:

- In the operation tree, right click on the "Status chart" folder and in the pop-up menu, select the command "Insert status chart".

or

- Open the window "Status chart" and call the "Edit" menu of right click and select the command "Insert contents" > "Table".

Switching between the status charts

After you have inserted a new status chart, a new tab is displayed at the lower edge of the window "Status chart":

If you wish to switch between the status tables:

- Click on the tab of the required status chart.

Displaying hidden tabs

Sometimes, a tab is hidden by the buttons on the righthand side that are used to scroll. If a tab cannot be seen, proceed as follows:

- Drag the demarcation line between the tab area and the scroll buttons to display additional tabs.

13.4.3.4 Creating a status chart

You can enter addresses in a status chart in order to monitor and control values from your program. Values of timers and counters can be displayed as bits or words. If you wish to display the value of a timer or a counter, then the state of the output is displayed (on or off). If you wish to display the value of a timer or a counter as word, then the actual value used.

Procedure

To create a status chart, proceed as follows:

1. Enter the addresses of the required value in the "Address" column.

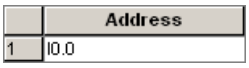
   All memory types are valid with the exception of accumulators and data constants.

   To edit an address field:

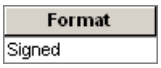
   – Select the required field using the cursor keys or the mouse.

   – If you enter data, existing data are deleted and the new characters are entered.

   – The field is selected if you double click with the mouse or press key <F2>. You can then move the cursor using the cursor keys to the position that you wish to edit.

2. If the element involves a bit (e.g. I, Q or M), then the bit format is displayed in the second column. If the element involves a byte, word or double word, select the field in the column "Format" and double click or press the space bar or ENTER in order to scroll through the valid formats until the correct format is displayed.

Note

You can select addresses in the symbol table and copy these into the status chart in order to more quickly generate your table.

You can display the status a multiple number of times. You can classify the elements in logical groups to display each group in an individual table. In this way, you avoid having to scroll through extremely long lists.

13.4.3.5 Editing the status chart

Displays

To scroll through the possible data formats for a specific address:

- Select the field "Data format" and repeatedly press the enter key.

To display all available data formats:

- Open the drop-down list field.

Changing

To set the width of a column:

- Position the mouse pointer at the edge of a column until the appearance of the cursor changes and then drag to increase or decrease the width of the column.

Selecting

To select a complete row (to cut or copy):

- Click the number of the row once.

To select the complete status chart:

- Click the upper lefthand corner once above the row numbers.

Inserting

To insert a new row:

1. Select a field or a row in the status chart
2. Open the menu "Edit" or click with the right mouse button on the field (in order that the contact menu is displayed).
3. Select the command "Insert contents" > "Row".

The new row is inserted in the status chart above the cursor position. The subsequent rows are shifted downwards by one row.

To insert a new row with the following address and the same data format:

- Select an address field and press the enter key.

To insert a row at the lower end of the status chart:

- Locate the cursor in a field in the last row and press the <arrow downwards> key.

Delete

To delete a field or a row:

1. Select the field or the row and click with the right mouse button.
2. Select the menu command "Delete" > "Selection"

If you delete a row, then the following rows shift upwards by one row.

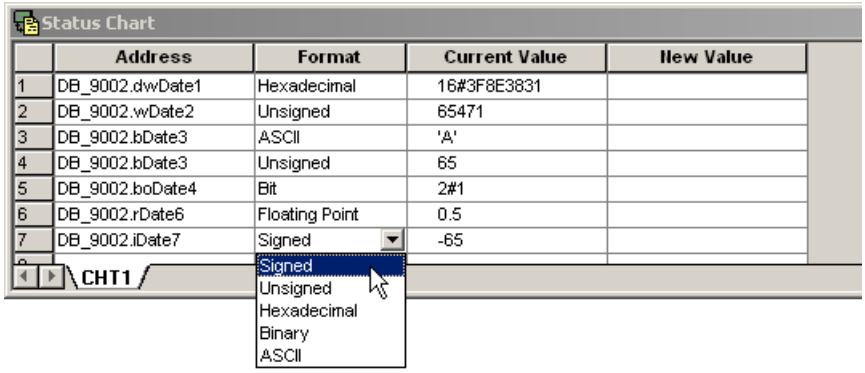
Navigating

To jump into the next field of the table:

- Press the <TAB> key.

13.4.3.6 Data formats

The data format that you assign to a value defines how the value can be represented in the status chart.



DB9002 is shown symbolically addressed in the diagram. The possible display formats for variables, type INTEGER or WORD are shown using an example of block variable iData7.

Note

Bit and binary values are both introduced by the number 2 and the # symbol.

Hexadecimal values are introduced by the number 16 and the # symbol.

Bit values have one digit. Binary values have eight digits.

Signed and unsigned values use the basis 10 (decimal).

13.4.3.7 Enabling the status table

Enable the status chart so that the status information can be updated.

Procedure

If you wish to continually update the status information in the status chart, enable the chart status:

- To do this, select the menu command "Test" > "Chart status".

or

- Click on the appropriate button in the function bar.

If you only require a "snapshot" of the values, execute the function "Single read":

- Select the menu command "Test" > "Single read".

or

- Click on the appropriate button in the function bar.

Note

When the chart status is enabled, then the "Single read" function is deactivated.

Note

If the chart is still empty, then enabling the status chart has not effect: You must first create your status chart by entering program values (addresses) in the "Address" column and you must enter a data type in the "Format" column for each address (see "Creating a status chart (Page 1252)" and "Data formats (Page 1255)").

13.4.3.8 Working with test functions in the status chart

You access the test functions (Single read, Write all,) using the menu "Test" or using the function bar with the test functions.

Single read

Use single read if you require a "snapshot", i.e. a single update of the program status of all values.

As default, the chart status continually interrogates the target system for status updates. If you click on a status chart and the chart status is disabled, then the button for single read is activated.

Write all

After you have entered the values in the column "New value" in the status chart, write the required changes to the target system using the command "Write all".

13.4.4 Execute cycles

You can specify that the target system should process a certain number of cycles of your program (from 1 cycle up to 65535 cycles). If you specify that the target system should execute a certain number of cycles, then you can monitor the processing of the process variables.

In the first cycle, the value of SM0.1 = 1 (ON).

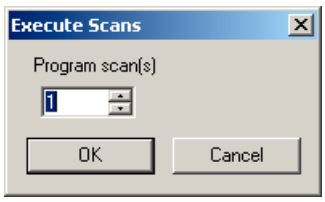
Executing a single cycle

1. The target system must be in STOP operating state. If it is not already in STOP, switch the target system into the STOP operating state.
2. Select the menu command "Test" > "First cycle".

Executing several cycles

1. The target system must be in STOP operating state. If it is not already in STOP, switch the target system into the STOP operating state.
2. To execute several cycles, select the menu command "Test" > "Several cycles".

   This opens the dialog box "Execute cycles":



3. Specify how many cycles should be executed, and confirm with "OK".

Note

Ensure that you switch the target system back into the RUN operating state if you wish to return to normal program processing:

- To do this, press the button "RUN" in the function bar.

or

- Select the menu command "Target system" > "RUN" window.

13.5 Data interface

Data is cyclically exchanged on one hand between the PLC and NC and and on the other hand between the PLC and HMI. This especially means that the data received from HMI and destined for the NC must be marshaled by the user program in order that these become effective.

Data to the PLC are provided by the firmware at the start of the user program cycle. This ensures, for example, that the signals from the NCK remain constant throughout a cycle.

Data from the PLC are transferred by the firmware to the NKC or HMI at the end of the user program cycle.

All data of this interface are listed in the manual for SINUMERIK 828D, PPU.

13.5.1 PLC-NCK interface

These cyclic data include, e.g. status signals ("Program running", "Program interrupted"), control signals (Start, Stop) and auxiliary and G functions.

Data are structured in signals for:

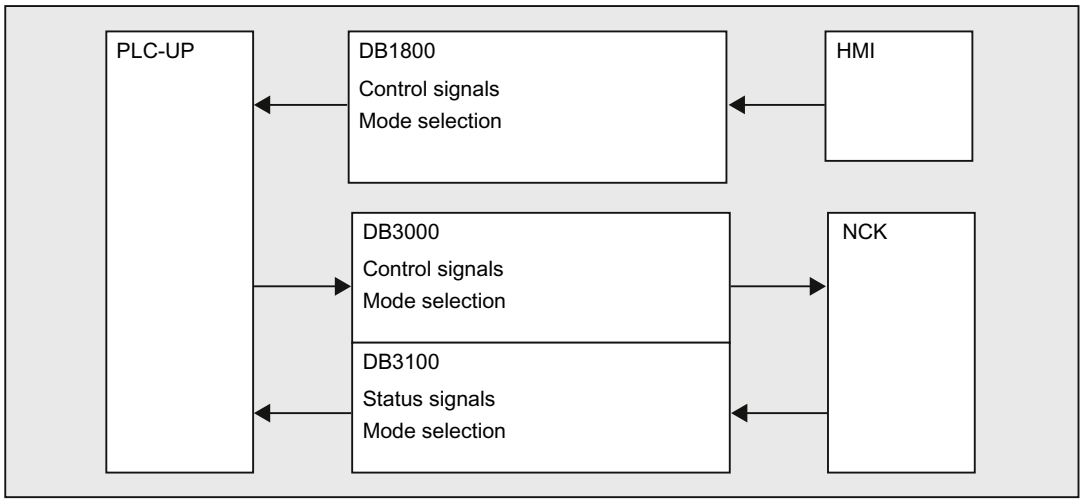
- Mode signals
- NC channel signals
- Axis and spindle signals
- General NCK signals
- Fast data exchange PLC-NCK

13.5.1.1 Mode signals

DB3000, 3100

The mode signals specified by the machine control panel or the HMI are transferred to the NCK.

There actual states are signaled to the PLC from the NCK.



13.5.1.2 NC channel signals

DB2500, 3200, 3300, 3500

The signals are structured as follows:

- Control/status signals with normal cyclic transfer, see "Mode signals (Page 1259) ".
- Auxiliary and G functions

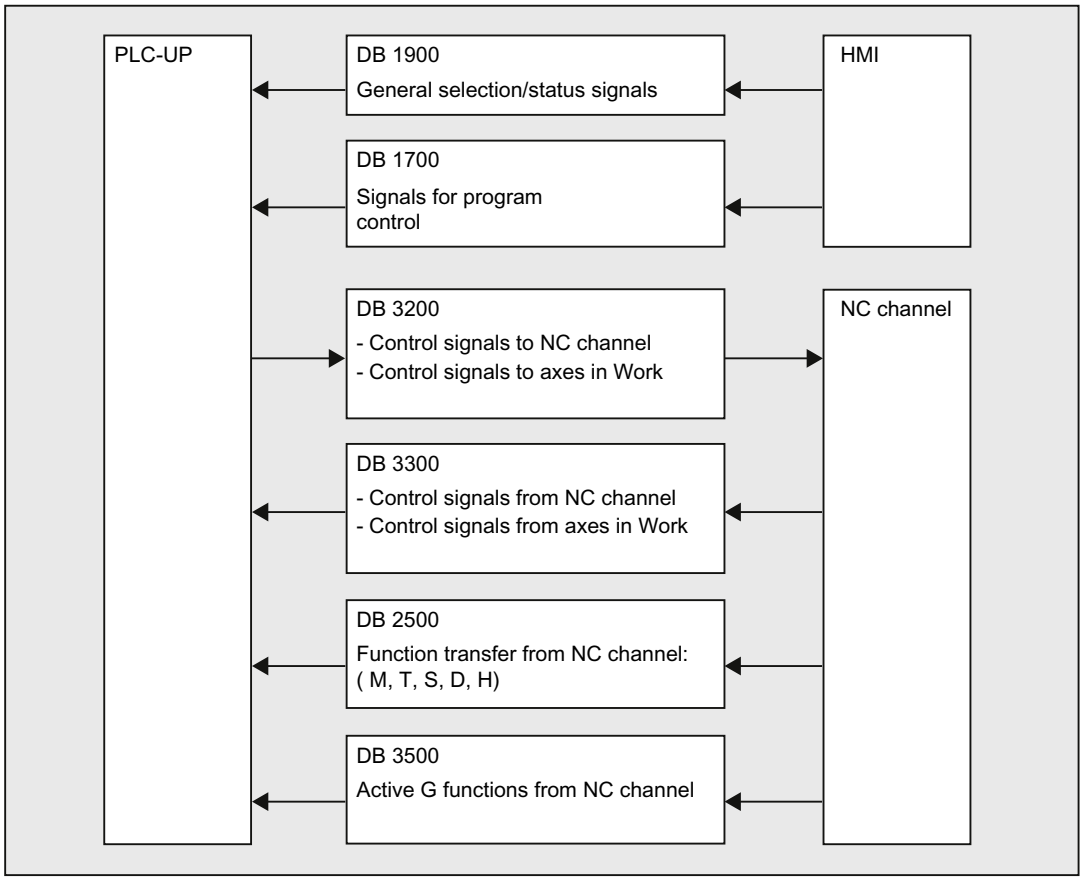
  These are entered in the interface DBs in two ways.

  First, they are entered with the change signals.

  The M signals M0 to M99 are additionally decoded and the associated interface bits are set for one cycle.

  For G commands, only the groups selected via machine data are entered in the interface data block.

  The S values are also entered together with the related M signals (M03, M04, M05) in the spindlespecific interface. The axisspecific feedrates are also entered in the appropriate axisspecific interface.



13.5.1.3 Axis and spindle signals

DB3200, 3300, 3700, 3800, 3900

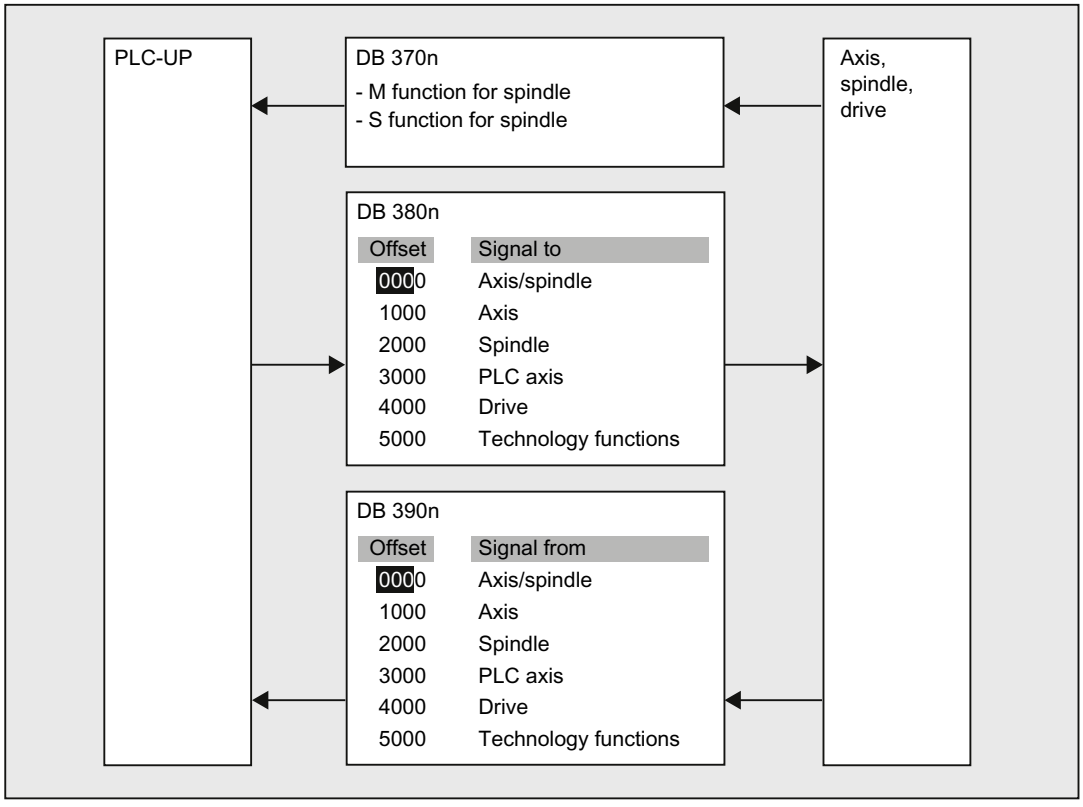
The axis-specific and spindle-specific signals are divided into the following groups:

- Shared axis/spindle signals
- Axis signals
- Spindle signals
- Drive signals

The signals are transferred cyclically with the following exceptions. The exceptions include axial F value, M and S value.

An axial F value is entered via the M, S, F distributor if it is transferred to the PLC during the NC machining process.

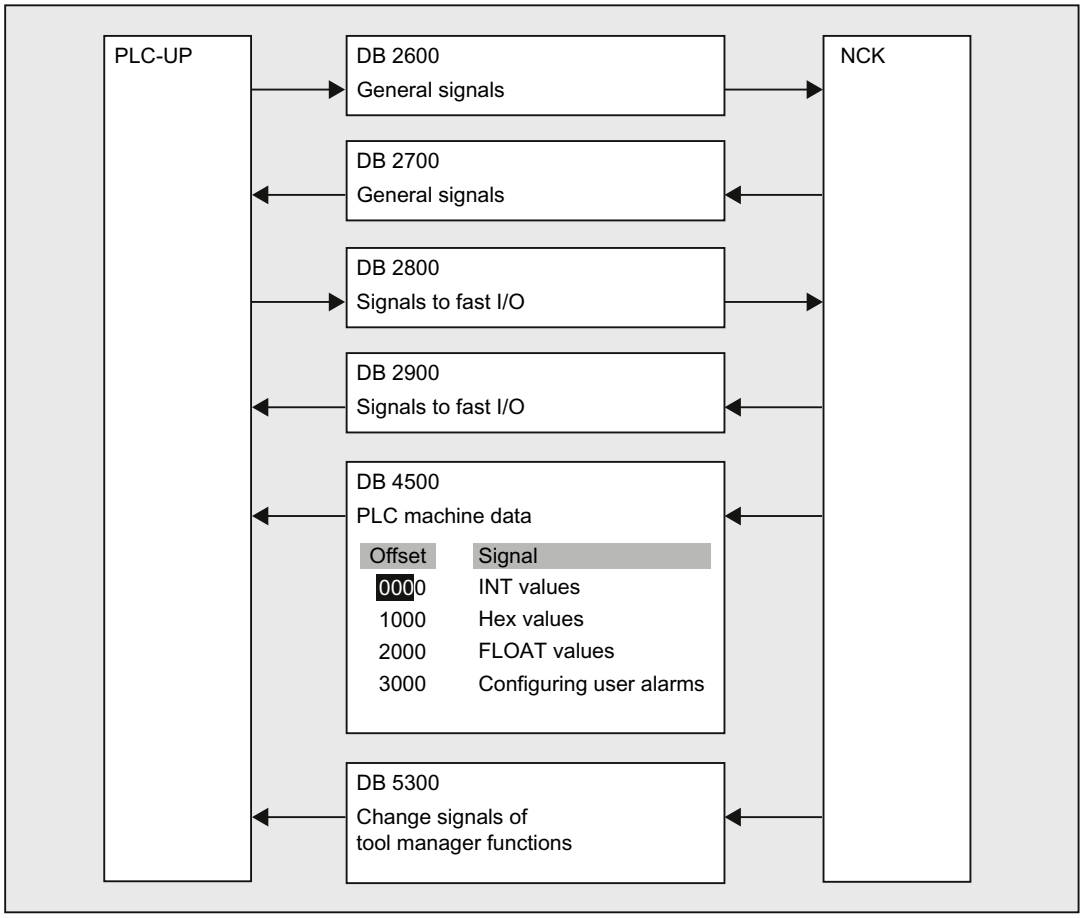
The M and S value are also entered via the M, S, F distributor if one or both values requires processing.



13.5.1.4 General NCK signals

DB2600, 2700, 2800, 2900, 4500, 5300

- Setpoints to digital/analog inputs/outputs of the NCK
- Actual values from the digital/analog inputs/outputs of the NCK
- Keyswitch and Emergency Stop signals
- Ready and status signals of the NCK



13.5.1.5 Fast data exchange PLC-NCK

DB4900

Data block DB4900 with a size of 1024 bytes is used for fast information exchange between the PLC and NCK.

The assignment of the area (structure) must be identically negotiated in the NC part program and PLC user user program.

This data can be accessed from the NC part program using the commands `$A_DBB[x]`, `$A_DBW[x]`, `$A_DBD[x]` and `$A_DBR[x]; 0 ≤ x ≤ 1023` (see Parameter Manual, System variables).

In this case, the alignment of the data must be selected corresponding to its format, i.e. a Dword starts at a 4byte limit and a word at a 2byte limit. Bytes can be located at any offset within the data field.

Data consistency is guaranteed for byte, word and Dword accesses. When transferring several data, the consistency must be guaranteed on the user-side using semaphores, which can be used to detect the validity or consistency of a block.

13.5.2 PLC-HMI interface

DB1700, 1800, 1900

These signals have already been specified in the diagrams of Chapter PLC-NCK interface (Page 1258).

A reference is again made to what has been stated under Data interface (Page 1258):

Data received from the HMI and destined for the NC are not automatically entered into the NC interface range. In fact, these signals and data must be marshaled by the user program.

It involves the following signals:

- Program selection via lists
- Messenger control command
- General signals from/to HMI
- Signals from/to the maintenance planner
- Signals from operator panel (retentive area)
- General selection/status signals from/to HMI (retentive area)

13.6 Function interface

13.6.1 Read/write NC variables

13.6.1.1 User interface

The PLC user program can read or write a maximum of eight NC variables simultaneously via the NC/PLC interface "Read/write NC variable".

The following steps must be performed as part of a job (read/write):

1. Job specification (Page 1264)
2. Job management: Start job (Page 1266)
3. Job management: Waiting for end of job (Page 1267)
4. Job management: Job completion (Page 1267)
5. Job evaluation (Page 1269)

Flow diagram of a job: See "Job management: Flow diagram (Page 1268)"

13.6.1.2 Job specification

Each variable that is to be processed in a job, must be specified in the **variable-specific job interface** via its parameters. The general identifiers are discussed in more detail later for each variable that can be accessed from the interface.

DB120x [1)]	Reading/writing NC data (PLC → NCK)							
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBB1000	Variable index							
DBB1001	Area number							
DBW1002	Column index, NCK variable x							
DBW1004	Line index, NCK variable x							
DBW1006	-							
DBD1008	Write: Data to NCK variable x (data type of the variables: 1…4 bytes)							
1) DB120**x**, with x = 0 ... 7 corresponds to variable 1 ... 8.								

NOTICE

Drive-specific variables

When reading/writing drive-specific variables, only the variables of exactly one SERVO drive object may be addressed in a job. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.

Error case

In the event of an error, reading/writing variables from different drive objects, or simultaneously from a channel and a drive object, an error message is output:

DB1200.DBX3000.1 == 1 (error occurred)

Example: Reading a variable of the "Location type" as the fourth variable

```
DB120x.DBB1000: 7
DB120x.DBB1001: -
DB120x.DBW1002: <Location number>
DB120x.DBW1004: <Magazine number>
DB120x.DBW1006: -
DB120x.DBD1008: -
```

Example: Writing a variable as the fourth variable

To write a data item to the NC, the value must be entered into the double word `DBD1008`:

```
DB120x.DBB1000: <Variable index>
DB120x.DBB1001: <Area number>
DB120x.DBW1002: <Column index>
DB120x.DBW1004: <Line index>
DB120x.DBW1006: -
DB120x.DBD1008: <value>
```

13.6.1.3 Job management: Start job

The following data must be written by the user to the **global job interface**:

DB120x [1]	Reading/writing NC data (PLC → NCK)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB 0							Job type	Job: Start
DBB 1	Number of variables to be processed in the job							
1) DB120**x**, with x = 0 ... 7 corresponds to variable 1 ... 8.								

Job type

- **Read** variable: DB1200.DBX0.1 = **0**
- **Write** variable: DB1200.DBX0.1 = **1**

Start job

The start signal must be set to start the job via a specified number of variables:

DB1200.DBX0.0 = **1**

Note

A new job can only be started if the previous job was completed. See Section "Job management: Waiting for end of job (Page 1267)".

The execution of a job may take several PLC cycles and vary depending on the utilization. Therefore, the time for this function cannot be defined.

13.6.1.4 Job management: Waiting for end of job

The end of the job is always signaled back by the NC for the **whole** job in the **global event interface**. The signals can only be read by the PLC user.

DB120x [1]	Reading/writing NC data (NCK -> PLC)							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB 2000							Error in job	Job completed
1) DB120**x**, with x = 0 ... 7 corresponds to variable 1 ... 8.								

Job status

- End of job without error

  DB1200.DBX2000.0 == **1 AND** DB1200.DBX2000.1 == **0**
- End of job with error

  DB1200.DBX2000.0 == **1 AND** DB1200.DBX2000.1 == **1**

Possible error causes

- Number of variables (DB1200.DBB1) out of the valid range
- Variable index (DB1200.DBB1000) out of the valid range
- Simultaneous reading/writing of NC data from different servo drive objects

13.6.1.5 Job management: Job completion

Requirement

In order to complete the job, the start signal of the job must be reset from the PLC user program after detection of the end of the job:

DB1200.DBX0.1 = **0**

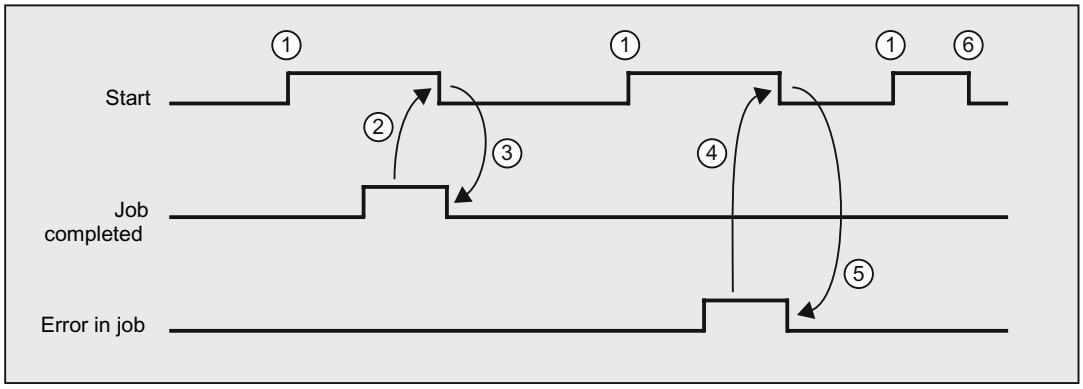
Feedback signal

As feedback, the NC resets the status signals:

- DB1200.DBX2000.0 == **0**
- DB1200.DBX2000.1 == **0**

The job is now completed.

13.6.1.6 Job management: Flow diagram



① Start job:
DB1200.DBX0.0 (start) = 1

② Waiting for end of job:
DB1200.DBX2000.0 (job completed) == **1** AND
DB1200.DBX2000.1 (error in job) == **0**

⇒ Reset job request:
DB1200.DBX0.0 = 0 (start)

③ With DB1200.DBX0 0 == 0 (start), the job is completed by the basic PLC program:
DB1200.DBX2000.0 (job completed) = 0

④ Waiting for end of job:
DB1200.DBX2000.0 (job completed) == **0** AND
DB1200.DBX2000.1 (error in job) == **1**

⇒ Perform error handling

⇒ Reset job request:
DB1200.DBX0.0 (start) = 0

⑤ With DB1200.DBX0.0 == 0 (start), the job is completed by the basic PLC program:
DB1200.DBX2000.1 (error in job) = 0

⑥ If DB1200.DBX0.0 (start) is reset before the end of job is signaled by the basic PLC program, the job is executed without further feedback.

13.6.1.7 Job evaluation

The variable-specific result interface must be evaluated for each variable processed in the job.

DB120x [1]	NC services (NC → PLC)							
DBB3000	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
							Error has occurred	Variable valid
DBB3001	Access result							
DBW3002								
DBD3004	Read value of the variable							
1) DB120**x**, with x = 0 ... 7 corresponds to variable 1 ... 8.								

Job status

Job without error

- DB1200.DBX3000.0 == 1 (variable valid) **AND**
- DB1200.DBX3000.1 == 0 (no error occurred)

Result:

- DB1200.DBB3001 == 0 (access result: "No error")
- DB1200.DBD3004 == <read value>

Job with error

- DB1200.DBX3000.0 == 0 (variable not valid) **AND**
- DB1200.DBX3000.1 == 1 (error occurred)

Result:

- DB1200.DBB3001: For possible access results, see below

Access result: DBB3001

Value	Meaning
0	No error
3	Access to object is not permitted
5	Invalid address
10	Object does not exist

13.6.1.8 Operable variables

Variable cuttEdgeParam

Compensation value parameters and cutting edge list with D numbers for a tool

The meanings of the individual parameters depend on the type of the tool in question. Currently, 25 parameters are reserved for each tool edge (but only a part of them is loaded with values). To be able to remain flexible for future extensions, it is not recommended to use a fixed value of 25 parameters for calculation, but the variable value 'numCuttEdgeParams' (variable index 2).

For a detailed description of tool parameters, refer to Chapter W1: Tool offset (Page 1567).

	Variable cuttEdgeParam [r/w]
DB120x.DBB1000	1
DB120x.DBB1001	-
DB120x.DBW1002	(Cutting edge No. - 1) * numCuttEdgeParams + ParameterNr (WORD)
DB120x.DBW1004	T number (1...32000) (WORD)
DB120x.DBD1008	Write: Data to NCK variable x (data type of the variables: REAL)
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable numCuttEdgeParams

Number of P elements of a cutting edge

	Variable numCuttEdgeParams [r]
DB120x.DBB1000	2
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: WORD)

Variable linShift

Translation of a settable work offset (channel-specific settable frames)

They only exist if MD18601 MM_NUM_GLOBAL_USER_FRAMES > 0.

There are the frame indices:

0: ACTFRAME = actual resulting work offset
1: IFRAME = actual settable work offset
2: PFRAME = actual programmable work offset
3: EXTFRAME = actual external work offset
4: TOTFRAME = actual total work offset = total of ACTFRAME and EXTFRAME
5: ACTBFRAME = actual total base frame
6: SETFRAME = actual 1st system frame (PRESET, scratching)
7: EXTSFRAME = actual 2nd system frame (PRESET, scratching)
8: PARTFRAME = actual 3rd system frame (TCARR and PAROT with orientable tool carrier)
9: TOOLFRAME = actual 4th system frame (TOROT and TOFRAME)
10: MEASFRAME = result frame for workpiece and tool gauging
11: WPFRAME = actual 5th system frame (workpiece reference points)
12: CYCFRAME = actual 6th system frame (cycles)

The max. frame index is 12.

The value of numMachAxes is contained in the variable with variable index 4.

	Variable linShift [r]
DB120x.DBB1000	3
DB120x.DBB1001	-
DB120x.DBW1002	Frame index * numMachAxes + axis number
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable numMachAxes

Number of the highest existing channel axis

If there are no gaps between channels, this corresponds to the number of existing axes in the channel.

	Variable numMachAxes [r]
DB120x.DBB1000	4
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: WORD)

Variable rpa

R-variables

	Variable rpa [r/w]
DB120x.DBB1000	5
DB120x.DBB1001	-
DB120x.DBW1002	R number + 1
DB120x.DBW1004	-
DB120x.DBD1008	Write: Data to NCK variable x (data type of the variables: REAL)
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable actLineNumber

Line number of the actual NC block (from 1)

0: Prior to program start
-1: Not available due to error
-2: Not available due to DISPLOF

	Variable actLineNumber [r]
DB120x.DBB1000	6
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: INT)

Variable magazine data: Location data

Location type ($TC_MPP2)

Location type: $TC_MPP2			
Address	**Description**	**Valid values**	
DB120x.DBB1000	Variables index	7	
DB120x.DBB1001	-	-	
DB120x.DBW1002	Location number	1 … 31999	
DB120x.DBW1004	Magazine number	1 … 9999	
DB120x.DBD1008	-	-	
DB120x.DBW3004	Result: Value of the NCK variable x	> 0	Location type for virtual location
	Data type: WORD	0	"match all" (buffer)
		9999	undefined (no virtual location)

Location status ($TC_MPP4)

Location state: $TC_MPP4			
Address	Description	Valid values	
DB120x.DBB1000	Variables index	8	
DB120x.DBB1001	-	-	
DB120x.DBW1002	Location number	1 … 31999	
DB120x.DBW1004	Magazine number	1 … 9999	
DB120x.DBD1008	-	-	
DB120x.DBW3004	Result: Value of the NCK variable x	1	Blocked
		2	free (<> occupied)
		4	reserved for tool in buffer
		8	reserved for tool to be loaded
		16	occupied in left half location
		32	occupied in right half location
		64	occupied in upper half location
		128	occupied in lower half location

T No. of tool at this location ($TC_MPP6)

T number of the tool at this location: $TC_MPP6		
Address	Description	Valid values
DB120x.DBB1000	Variables index	9
DB120x.DBB1001	-	-
DB120x.DBW1002	Location number	1 … 31999
DB120x.DBW1004	Magazine number	1 … 9999
DB120x.DBD1008	-	-
DB120x.DBW3004	Result: T number of the tool at this location	T number of the tool

Variable r0078[1]

CO: Current actual value, torque-generating [Arms]

Index: [1] = Smoothed with p0045

	Variable r0078[0...1] [r]
DB120x.DBB1000	10
DB120x.DBB1001	Number of the drive module
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable r0079[1]

CO: Torque setpoint at the output of the speed controller (before clock cycle interpolation) [Nm]

Index: [1] = Smoothed with p0045

	Variable r0079[0...1] [r]
DB120x.DBB1000	11
DB120x.DBB1001	Number of the drive module
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable r0081

CO: Torque utilization in percent

	Variable r0081 [r]
DB120x.DBB1000	12
DB120x.DBB1001	Number of the drive module
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable r0082[1]

CO: Active power actual value [kW]

Index: [1] = Smoothed with p0045

	Variable r0082[0…2] [r]
DB120x.DBB1000	13
DB120x.DBB1001	Number of the drive module
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variables: Temperature compensation

Variable TEMP_COMP_ABS_VALUE (SD43900)

Position-independent temperature compensation value

	Variable TEMP_COMP_ABS_VALUE [r/w]
DB120x.DBB1000	14
DB120x.DBB1001	No. of the axis (1, 2, ...)
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	Write: Data to NCK variable x (data type of the variables: REAL)
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable TEMP_COMP_SLOPE (SD43910)

Gradient for position-dependent temperature compensation

	Variable TEMP_COMP_SLOPE [r/w]
DB120x.DBB1000	15
DB120x.DBB1001	No. of the axis (1, 2, ...)
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	Write: Data to NCK variable x (data type of the variables: REAL)
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable TEMP_COMP_REF_POSITION (SD43920)

Reference position for position-dependent temperature compensation

	Variable TEMP_COMP_REF_POSITION [r/w]
DB120x.DBB1000	16
DB120x.DBB1001	No. of the axis (1, 2, ...)
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	Write: Data to NCK variable x (data type of the variables: REAL)
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable TOOL_TEMP_COMP (SD42960[...])

Temperature compensation referred to the tool

	Variable TEMP_COMP_REF_POSITION [r/w]
DB120x.DBB1000	17
DB120x.DBB1001	-
DB120x.DBW1002	Index + 1 (1, 2, 3)
DB120x.DBW1004	-
DB120x.DBD1008	Write: Data to NCK variable x (data type of the variables: REAL)
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: REAL)

13.6.2 Program instance services (PI services)

13.6.2.1 User interface

Job specification

PI services are specified via their **job interface** (DB1200 from offset 4000) via their parameter.

DB1200			PI service [r/w]					
			PLC -> NCK interface					
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBB 4000								Start
DBB 4001	PI index							
DBB 4002								
DBB 4003								
DBW 4004	PI parameter 1							
DBW 4006	PI parameter 2							
DBW 4008	PI parameter 3							
DBW 4010	PI parameter 4							
DBW 4012	PI parameter 5							
DBW 4014	PI parameter 6							
DBW 4016	PI parameter 7							
DBW 4018	PI parameter 8							
DBW 4020	PI parameter 9							
DBW 4022	PI parameter 10							

PI index

DB1200.DBB4001 specifies the specific PI service.

PI parameter n

From DB1200.DBW4004, PI parameter n must be specified for the specific service.

Start

DB1200.DBX4000.1 = 1: Job is started for the specified number of variables.

Job feedback

The PLC firmware provides feedback as to whether the started PI service was successful or not successful in the **result interface** (DB1200 from offset 5000).

The job end is signaled using one of two signals:

DB1200.DBX5000.0 == 1 or DB1200.DBX5000.1 == 1

The signals are written by the PLC operating system; therefore, they can only be read by the user. A job has been completed if both acknowledgement signals are zero. They become zero if the user resets the signal "Start" (DB1200.DBX4000.1) after the job end.

DB1200			PI service [r]					
			NCK -> PLC interface					
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBB 5000							Error in job	Job completed
DBB 5001								
DBB 5002								

Job completed

DB1200.DBX5000.0 = 1 job processing completed without error.

DB1200.DBX5000.0 = 0 otherwise is zero, if the user resets "Start".

Error in job

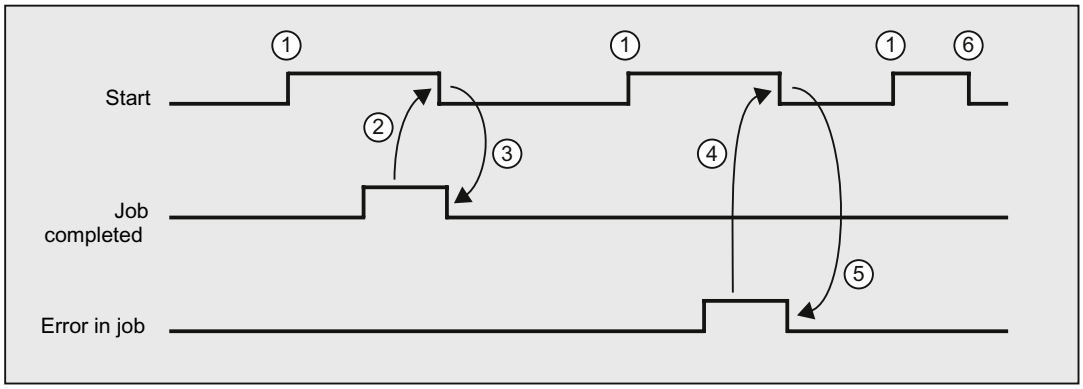
DB1200.DBX5000.1 = 1 job had an error, execution terminated.

DB1200.DBX5000.1 = 0 otherwise is zero, if the user resets "Start".

Possible error causes

- Index of the PI service (DB1200.DBB4001) outside the valid range
- Parameter error

PI services: Cycle diagram



① User sets the signal "Start", job execution starts.
② After the PLC firmware signals "Job completed", the user resets the signal "Start".
③ By resetting the signal "Start", the PLC firmware resets the signal "Job completed".
④ After the PLC firmware signals "Error in job", the user resets the signal "Start".
⑤ By resetting the signal "Start", the PLC firmware resets the signal "Error in job".
⑥ If the user accidentally resets the signal "Start" before one of the signals "Job completed" or "Error in job" is received, then the result signals for this job are not updated. However, the job is executed.

13.6.2.2 PI services

PI service ASUB

Function

With the "ASUB" PI service, it is possible to assign one program each to the interrupt numbers 1 and 2 from the PLC. These programs must be present in the NC in the machine manufacturer directory (CMA) with fixed program names:

Interrupt:	Directory	Program name
1	_N_CMA_DIR	PLCASUP1_SPF
2	_N_CMA_DIR	PLCASUP2_SPF

If the above mentioned programs are not available, they must be created in the machine manufacturer directory (CMA) of the NC. A power on reset must then be performed on the NC.

The PI service must only be executed once after a restart. The assignment of the interrupt to the program is retained.

PI service: ASUB		
Address	Description	Valid values
DB1200.DBW4001	PI index	1, 2
DB1200.DBW4004	LIFTFAST	0, 1
DB1200.DBW4006	BLSYNC	0, 1

PI index	Function	Interrupt priority
1	Assignment: Interrupt 1 to _N_CMA_DIR/PLCASUP1_SPF	1
2	Assignment: Interrupt 2 to _N_CMA_DIR/PLCASUP2_SPF	2

Supplementary conditions

- The PI service may only be executed when the channel is in the RESET state.
- If a "run-up" is configured as the initiating event for the event-driven program call, the PI service must only be started after the end of the event-driven program.

PI service LOGOUT

Reset password

The password last transferred to the NCK is reset.

PI index	Function
3	Reset password

Relevant PI parameters for PI service 3

None

PI service _N_DASAVE

Service to safe data from SRAM to FLASH

PI index	Function
4	Data save from SRAM to FLASH

Relevant PI parameters for PI service 4

None

PI service TMMVTL

Function

Using the PI service TMMVTL, it is possible to initiate a job from the PLC to relocate a tool. After an error-free PI start, the TM executes an empty location search in the target magazine for the tool in the specified source location. The PLC then receives a job to relocate the tool via DB41xx.DBB0.

PI service: TMMVTL		
Address	Description	Valid values
DB1200.DBW4001	PI index	5
DB1200.DBW4004	Tool number	-1, 1 … 31999
DB1200.DBW4006	Source location number	-1, 1 … 31999
DB1200.DBW4008	Source magazine number	-1, 1 … 9999
DB1200.DBW4010	Target location number	-1, 1 … 31999
DB1200.DBW4012	Target magazine number	-1, 1 … 9999

- The tool can be specified either using a T number or by means of the location and magazine numbers. An unused specification has the value -1.
- With the target location number = -1, a search is made in the complete magazine for an empty location for the tool according to the search strategy that has been selected. If a target location is specified, then a check is made as to whether the location with the specified target location number is free and suitable for the particular tool.
- For a target magazine number = -1, a search is made in a buffer for the tool corresponding to the assignment obtained from $TC_MDP2.

Examples

- When using buffers to return the tool (for example Toolboy and/or shifter), an explicit empty location search in the magazine may be needed during the asynchronous return transport. In this case,the PLC does not have to note the original location, this PI service searches for a suitable location.
- A tool should be moved from a background magazine to the foreground magazine.

13.6.3 PLC user alarms

13.6.3.1 User interface

Note

Although the name user "*alarms*" is used in the following, it is only defined as to whether it involves a **message** or an **alarm** when entering the particular **cancel criterion** in bits 6 and 7 of machine data MD14516[x].

The user alarms are created in the HMI and are prepared for automatic processing. The user interface of the DB1600 permits:

- these 248 user alarms of numbers 700000 to 700247 to be activated,
- to be provided with an additional numerical parameter, and
- to be activated and acknowledged as well as
- to evaluate the system reactions initiated by it.

Activation interface of the user alarms

Each user alarm is activated using its assigned activation bit. These bits are set in the **activation interface** (DB1600 from offset 0):

A new user alarm is activated with a 0/1 edge of the particular bit.

DB1600			**Activating alarm [r/w]**					
Data block			PLC -> HMI interface					
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	Activation of Alarm No.							
DBB 0	700007	700006	700005	700004	700003	700002	700001	700000
	Activation of Alarm No.							
DBB 1	700015	700014	700013	700012	700011	700010	700009	700008
	Activation of Alarm No.							
DBB 2	700023	700022	700021	700020	700019	700018	700017	700016
	Activation of Alarm No.							
DBB 3	700031	700030	700029	700028	700027	700026	700025	700024
	Activation of Alarm No.							
DBB 4	700039	700038	700037	700036	700035	700034	700033	700032
	Activation of Alarm No.							
DBB 5	700047	700046	700045	700044	700043	700042	700041	700040
	….							
…								
	Activation of Alarm No.							
DBB 30	700247	700246	700245	700244	700243	700242	700241	700240

Variables interface of the user alarms

Each user alarm can be given a variable as parameter. One double word each is reserved for this purpose in the **variable interface** from offset 1000. As a consequence, valid offsets must be divisible by 4.

DB1600	Variable for alarm [r32/w32]
Data block	PLC -> HMI interface
DBD 1000	Variable for alarm 700000
DBD 1004	Variable for alarm 700001
DBD 1008	Variable for alarm 700002
	...
DBD 1980	Variable for alarm 700245
DBD 1984	Variable for alarm 700246
DBD 1988	Variable for alarm 700247

Alarm response and cancel criterion

Alarm response and cancel criterion

These two terms involve the conception and configuration of user alarms. The following attributes can be specified for each alarm:

- Alarm response: How the controller responds when an error occurs.
- Cancel criterion: What must be done to cancel the alarm again or acknowledge it. The cancel criterion simultaneously defines the alarm priority.

The setpoints are configured in the following machine data and their actual values are signaled together in one byte of the user interface: The setpoints are configured bit-coded in: MD14516[x], 0 ≤ x ≤ 247; whereby x = user alarm number - 700000

Active alarm response and cancel criterion

The present **active alarm responses** (i.e. the actual responses) and the **active cancel criteria** can be globally read out from the interface:

DB1600	Active alarm response [r]							
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBB 2000	POWER ON	Interrupt with DB1600 DBX3000.0		PLC STOP	Emergency stop	Feedrate disable for all axes	Read-in disable	NC start disable
DBB 2001								
DBB 2002								
DBB 2003								

One bit is set if for at least one active alarm the corresponding response or the corresponding cancel criterion is configured. It is canceled, if this response/cancel criterion is no longer configured for any of the alarms present.

The codings of the cancel criteria and the corresponding priorities are:

Bit 7	Bit 6	User alarm is acknowledged by	Type	Priority
0	0	Resetting of the activation bit	Message	Low
0	1	Acknowledgement in DB1600.DBX3000.0 (refer below)	Alarm	Medium
1	0	Power on	Alarm	High
1	1	Reserved (internally evaluated as {1, 0})		

Note

If none of bits 0 to 4 are set for an alarm/message in the machine data, then this defines that it involves a so-called "display message" that has no effect on the system. This especially indicates that also bits 6 and 7 of the corresponding machine data are evaluated.

Active alarm responses and cancel criteria of the user alarms

The present **active alarm responses** (i.e. the actual responses) and the **active cancel criteria** can be globally read-out of the interface.

DB1600	Active alarm response [r]							
Data block								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB 2000	POWER ON	Interrupt with DB1600 DBX3000.0		PLC STOP	Emergency stop	Feedrate disable for all axes	Read-in disable	NC start disable
DBB 2001								
DBB 2002								
DBB 2003								

One bit is set if for at least one active alarm the corresponding response or the corresponding cancel criterion is configured. It is canceled, if this response/cancel criterion is no longer configured for any of the alarms present.

Acknowledgement interface of the user alarms

Requirement

Requirement to acknowledge a user alarm is that the corresponding activation bit is reset.

- Messages with cancel criterion {0,0} then disappear automatically from the display.
- Alarms with cancel criterion {0,1} are canceled by the acknowledgement bit *Ack*.
- Alarms with cancel criterion {1,0} are not influenced when the acknowledgement bit is set and can only be canceled by a Power On.

DB1600	Alarm acknowledgement [r/w]							
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB3000								Ack
DBB3001								
DBB3002								
DBB3003								

Interface to HMI.

The PLC can transfer eight messages or alarms for display on the HMI, which are displayed in the sequence that they occur.

When additional messages/alarms occur, the first seven are kept in the HMI, and the latest message or the latest alarm is displaced from one that has just occurred according to the following rules:

- System message/alarm displaces user message/alarm
- Messages/alarms with a higher priority displace those of a lower priority.

The first seven messages/alarms are kept in the display because it is very probable that these define the cause of the problem and the following are just of a secondary nature. However, if one or several messages/alarms are acknowledged and therefore cleared, then a corresponding number of alarms/messages that have been received move up in the HMI.

13.6.4 PLC axis control

13.6.4.1 Overview

Target

The PLC can control eight axes or spindles via data blocks of the user interface; the axis/spindle is specified by its DB number:

- **DB380x** PLC → NCK interface
- **DB390x** NCK → PLC interface
- **DB390x** Axis index: 0 ≤ x ≤ 7, whereby axis index = axis number-1

The following functions are supported:

- Positioning axes
- Spindle positioning
- Rotate spindle
- Oscillate spindle
- Indexing axes

References:

- Function Manual, Extended Functions; Positioning Axes (P2) and Indexing axes (T1)
- Function Manual, Basic Functions; Spindles (S1)

Precondition

The axis to be controlled must be assigned to the PLC. An axis can be interchanged between NC and PLC using the user interface "*Axis interchange*" (DB3800.DBB8/DB3900.DBB8).

Function start

Each function is activated by the positive edge of the corresponding "Start" signal. This signal must remain a logical "1" until the function has been positively or negatively acknowledged (e.g. using *Position reached* = "1" or *Error* = "1"). The signal "Positioning axis active" = "1" indicates that the function is active and that the output signals are valid.

Interrupt

It is **not**possible to **interrupt** the function by resetting the start signal, but only via other interface signals (using the axis-specific signal *Delete distance to go/spindle reset*, DB380x DBX2.2).

The axis interface returns axis status signals that may need to be evaluated (e.g. *exact stop*, *travel command*, → DB390x).

If the axis/spindle is being traversed via the NC program when the PLC axis control is called (travel command present), then the function is only started after this movement has ended. No error code is output in this situation.

Axis disable

With the **axis disable** set, an axis controlled via PLC axis control will not move. Only a simulated actual value is generated. (Behavior as with NC programming).

13.6.4.2 User interface: Preparing the NC axis as PLC axis

Firstly, the axis/spindle must be requested from the PLC. This is realized via the following user interface:

Common signals to axis/spindle (excerpt)

Request axis or spindle:

DB3800 ... 3807			Signals to axis/spindle [r/w]					
Data block			PLC → NCK interface					
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB 0008	Request PLC axis/ spindle			Activation signal when this byte is changed				Request NC axis/ spindle

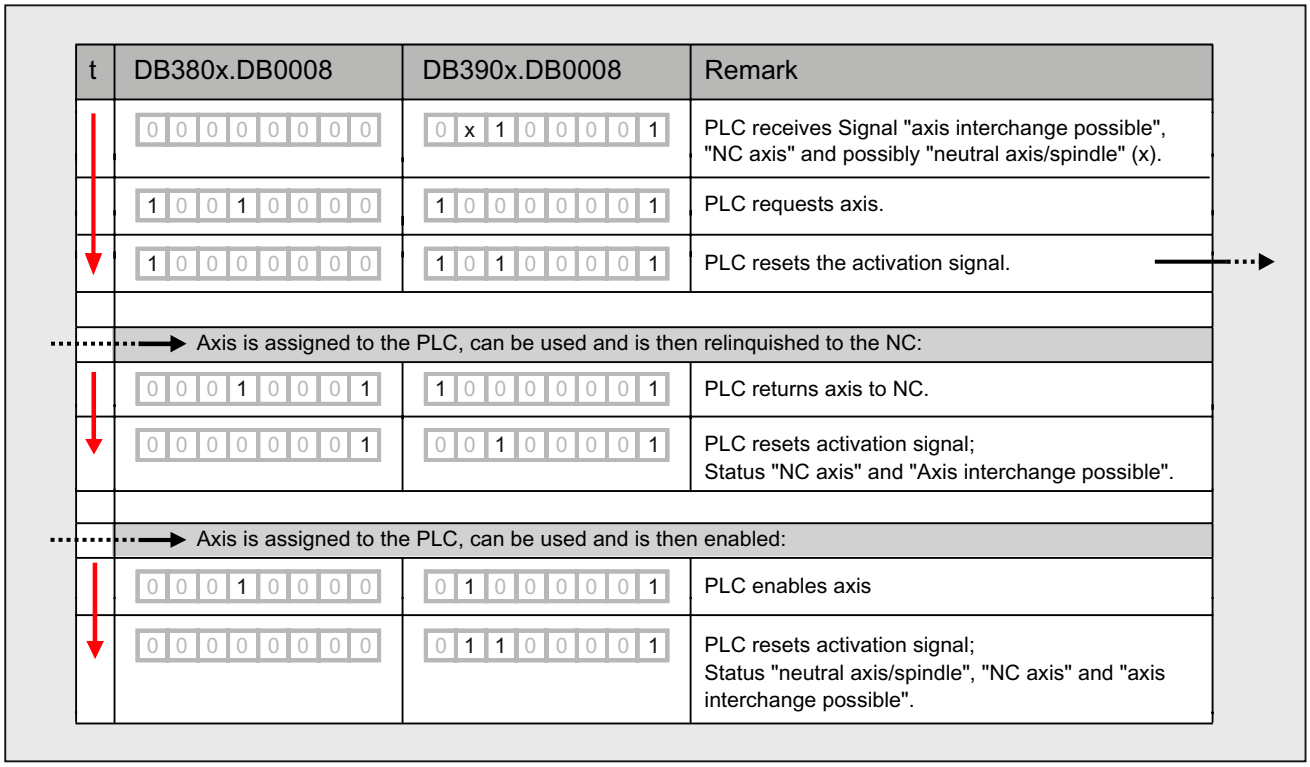
Every change to a request bit at the interface (DB380x.DBX0008.7 or bit DB380x.DBX0008.0) must be signaled to the NC using a 0→1 edge of the activation signal (bit DB380x.DBX0008.4). This activation signal should be reset again after one cycle.

General signals from axis/spindle (excerpt)

State interrogation is possible via the interface DB390x.DBB8. However, for simulation, the axis machine data MD30350 MA_SIMU_AX_VDI_OUTPUT must be set.

DB3900 ... 3907			Signals from axis/spindle [r]					
Data block			NCK → PLC interface					
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB 0008	PLC axis/ spindle	Neutral axis/ spindle	Axis interchange possible	New type requested from PLC				NC axis/ spindle

Request and relinquish PLC axis



t	DB380x.DB0008	DB390x.DB0008	Remark
	0 0 0 0 0 0 0 0	0 x 1 0 0 0 0 1	PLC receives Signal "axis interchange possible", "NC axis" and possibly "neutral axis/spindle" (x).
	1 0 0 1 0 0 0 0	1 0 0 0 0 0 0 1	PLC requests axis.
	1 0 0 0 0 0 0 0	1 0 1 0 0 0 0 1	PLC resets the activation signal.
	Axis is assigned to the PLC, can be used and is then relinquished to the NC:		
	0 0 0 1 0 0 0 1	1 0 0 0 0 0 0 1	PLC returns axis to NC.
	0 0 0 0 0 0 0 1	0 0 1 0 0 0 0 1	PLC resets activation signal; Status "NC axis" and "Axis interchange possible".
	Axis is assigned to the PLC, can be used and is then enabled:		
	0 0 0 1 0 0 0 0	0 1 0 0 0 0 0 1	PLC enables axis
	0 0 0 0 0 0 0 0	0 1 1 0 0 0 0 1	PLC resets activation signal; Status "neutral axis/spindle", "NC axis" and "axis interchange possible".

13.6.4.3 User interface: Functionality

The two tables provide an overview of the available interface signals. The precise description of the signals and the explanation of what signals are relevant for the individual functions are explained in the following.

Signals to PLC axis

DB3800...3807			Signals to PLC axis [r/w]					
Data block			PLC -> NCK interface					
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
DBB 3000	Start positioning axis	Start spindle positioning	Start spindle rotation	Start spindle oscillation				
DBB 3001			Stop spindle rotation	Stop spindle oscillation				
DBB 3002	Automatic gear selection	Constant cutting rate	Direction of rotation as for M4		Handwheel override	Traversing dimension, inches (not metric)	Distance condition, shortest distance (DC)	Distance condition, incremental (IC))
DBB 3003	Indexing position						Distance condition, abs. pos. direction (ACP)	Distance condition, abs. neg. direction (ACN)
DBD 3004	Position (REAL, with indexing axis: DINT)							
DBD 3008	Feedrate velocity (REAL), if < 0, the value is taken from machine data POS_AX_VELO							

The bits of the distance conditions and the direction of rotation definition define the particular positioning or traversing mode, only one of the bits must be set:

Meaning	Distance condition to be set
Positioning absolute	No mode bit set
Positioning incremental	DBB3002.0 = 1
Positioning, shortest distance	DBB3002.1 = 1
Positioning, absolute, positive approach direction	DBB3003.1 = 1
Positioning, absolute, negative approach direction	DBB3003.0 = 1
Direction of rotation as for M4	DBB3002.5 = 1

The remaining bits are used to specify and start the particular function, these function bits as well as position and velocity are explained in more detail for the individual functions.

Signals from PLC axis

3900...3907			Signals from PLC axis [r]					
Data block			PLC -> NCK interface					
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
3000	Positioning axes active	Position reached					Error while traversing	Axis cannot be started
3001								
3002								
3003	Error number							

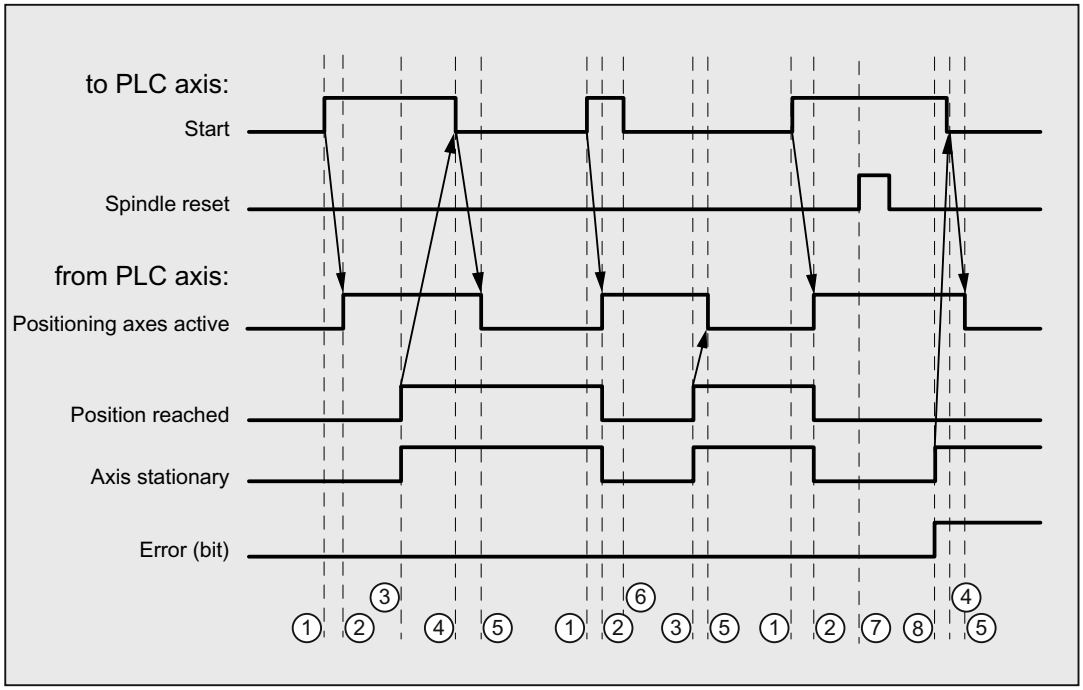
The following requirements must be satisfied in order to use the functions listed below:

- The axis or spindle is correctly assigned to the PLC.
- Controller and pulse enable are set.
- After setting all of the control signals, only one of the start signals is set in DB380x.DBB3000.

13.6.4.4 Spindle positioning

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0|1	Only one of the bits must be set, if all bits are 0, then this means absolute positioning.
DBX3002.1	Shortest path	0|1	
DBX3003.0	Absolute, negative direction	0|1	
DBX3003.1	Absolute, positive direction	0|1	
DBD3004	Setpoint position/setpoint distance	REAL	For "incremental": Setpoint distance
DBD3008	Feedrate	REAL	If = 0, the value from MD35300 $MA_SPIND_POSCTRL_VELO (position control activation speed) is taken
DBX3000.6	Start	0|1	Reset does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	0|1	Interrupt signal, exits the function

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	Also 1 when override = 0 or position setpoint reached when start = 1
DBX3000.6	Position reached	1: Position setpoint reached with "Exact stop fine"
DBX3000.0	Spindle cannot be started	
DBX3000.1	Error while traversing	1: Error during traversing, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	1: if n < n_{min}



① Function activated by user with a positive edge of *Start*.

② *Positioning axis active* message shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* may be withdrawn. For path specification = 0, the signals are not canceled.

③ When the position is reached this is signaled (*Position reached*), *Spindle stationary* is set.

④ The user then withdraws *Start*.

⑤ The *Positioning axis active* signal is then reset.

⑥ The user immediately resets the *Start* signal with receipt of the *Positioning axis active* signal.

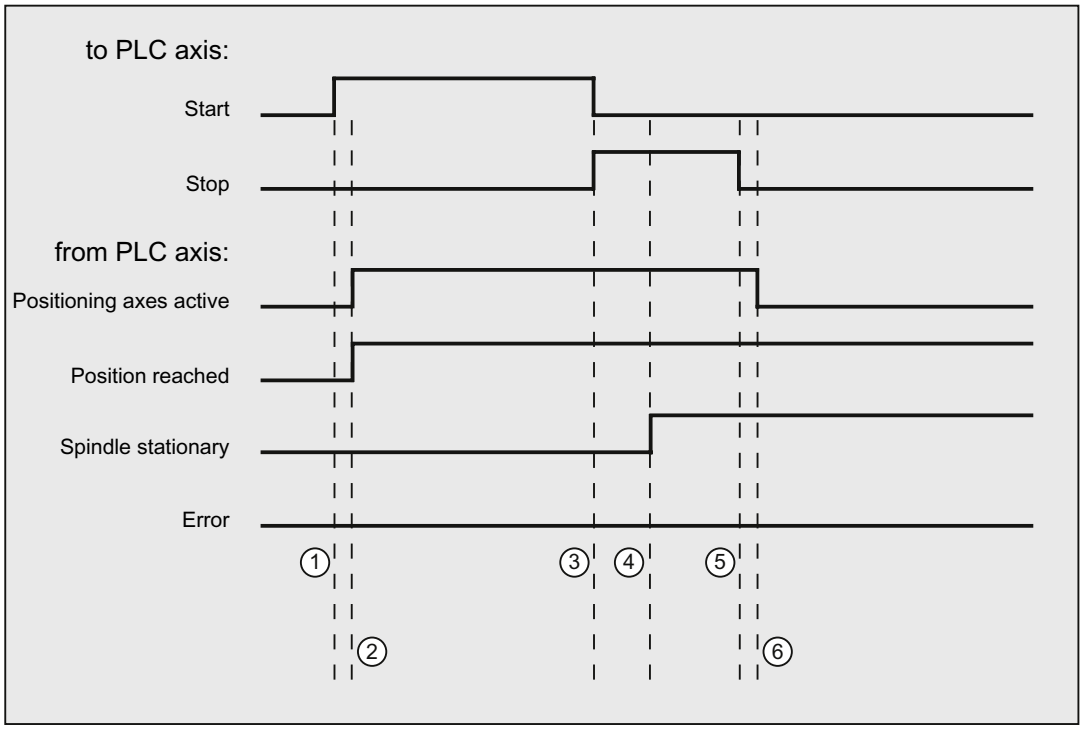
⑦ Positioning is aborted by setting *Spindle reset*. This signal must be present for at least one PLC cycle.

⑧ The spindle comes to a standstill (*Spindle stationary*), the *Fault* signal is set. (In this case, fault number 115 is output.)

13.6.4.5 Rotate spindle

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0	*Direction of rotation as for M4:*
DBX3002.1	Shortest distance	0	1: Direction of rotation specified by M4
DBX3002.5	Direction of rotation as for M4	0|1	0: Direction of rotation specified by M3
DBX3003.0	Absolute, negative direction	0	
DBX3003.1	Absolute, positive direction	0	
DBD3008	Feedrate velocity	REAL	Spindle speed
DBX3000.5	Start spindle rotation	0|1	
DBX3001.5	Stop spindle rotation	0|1	

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	1: for start or stop == 1
DBX3000.6	Position reached	1: Function was started without an error
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	



① Function activated by user with a positive edge of *Start*.

② Signals *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.

③ The user stops spindle rotation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

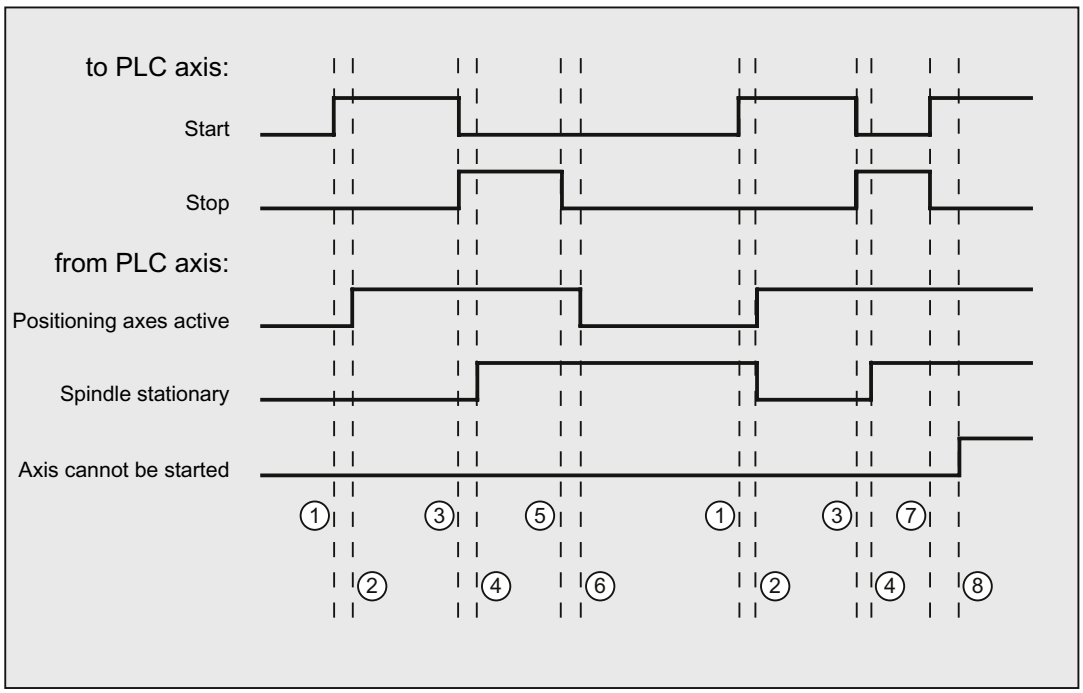
⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

13.6.4.6 Oscillate spindle

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0	It is not permissible that any of the bits are set.
DBX3002.1	Shortest distance	0	
DBX3002.5	Direction of rotation as for M4	0	
DBX3003.0	Absolute, negative direction	0	
DBX3003.1	Absolute, positive direction	0	
DBD3004	Setpoint gear stage: Values 0, 1, 2, 3, 4, 5	REAL	MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 0 0-5: Oscillation MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 1 0: Oscillation 1: Oscillation with gear stage change M41 2: Oscillation with gear stage change M42 3: Oscillation with gear stage change M43 4: Oscillation with gear stage change M44 5: Oscillation with gear stage change M45
DBD3008	Feedrate velocity	REAL	When oscillating, no significance! The oscillation speed is taken from machine data MD35400, $MA_SPIND_OSCILL_DES_VELO.
DBX3000.5	Start spindle oscillation	0|1	It is not permissible that the start directly follows a stop. Stop must first be reset (both 0).
DBX3001.5	Stop spindle oscillation	0|1	

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	1: for start or stop == 1
DBX3000.6	Position reached	1: after start
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBX3000.0	Axis cannot be started	1: Error when starting, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	



① Function activated by user with a positive edge of *Start*.
Note: This is only possible when the *Positioning axis active* signal is reset!

② Signals *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant and is therefore not shown.

③ The user stops spindle oscillation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

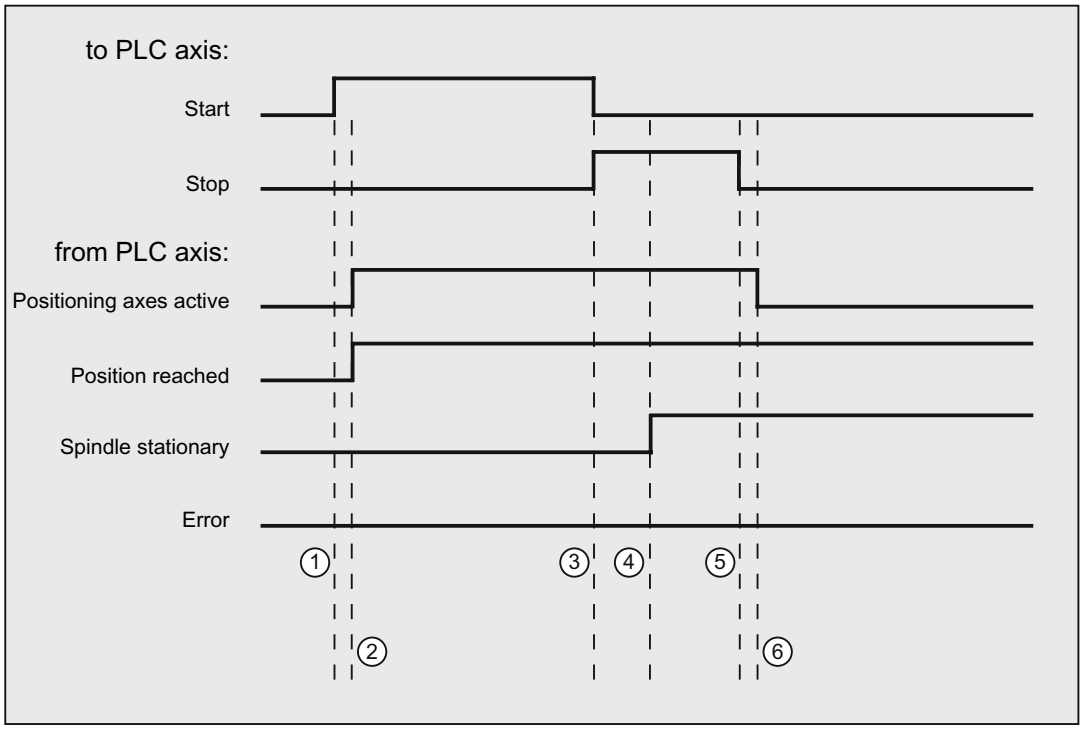
⑦ *Stop* is reset in the user program and *Start* is again set, incorrectly, in the same PLC cycle. This means that *Positioning axis active* is not reset, but...

⑧ ...the *Axis cannot be started* signal is set (error number 106).

13.6.4.7 Indexing axis

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0|1	Only one of the bits must be set, if all bits are 0, then this means absolute positioning.
DBX3002.1	Shortest distance	0|1	
DBX3002.5	Direction of rotation as for M4	0	
DBX3003.0	Absolute, negative direction	0|1	
DBX3003.1	Absolute, positive direction	0|1	
DBD3003.7	Indexing position	1	Indexing axis ON
DBD3004	Setpoint position/setpoint distance	DINT	for "incremental": Setpoint distance
DBD3008	Feedrate velocity	REAL	if = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).
DBX3000.7	Start positioning axis	0|1	Reset does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	0|1	Interrupt signal, exits the function

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	Also 1, if override = 0 or setpoint position reached.
DBX3000.6	Position reached	1: Setpoint position reached with "Exact stop fine".
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	



① Function activated by user with a positive edge of *Start*.
Note: This is only possible when the *Positioning axis active* signal is reset!

② Signals *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.

③ The user stops spindle oscillation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

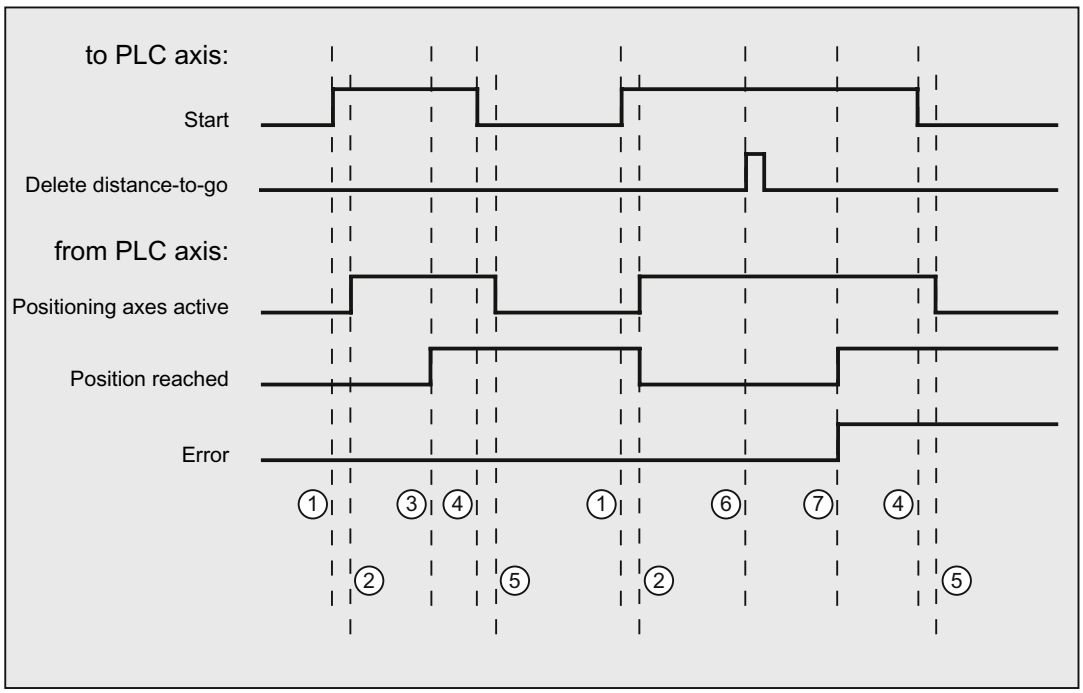
⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

13.6.4.8 Positioning axis metric

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0|1	Only one of the bits must be set, if all bits are 0, then this means absolute positioning.
DBX3002.1	Shortest distance	0|1	
DBX3002.5	Direction of rotation as for M4	0	
DBX3003.0	Absolute, negative direction	0|1	
DBX3003.1	Absolute, positive direction	0|1	
DBD3002.2	Traversing dimension inch	0	Traversing dimension, metric
DBD3002.3	Handwheel override	0	Override OFF
DBD3004	Setpoint position/setpoint distance	REAL	for "incremental": Setpoint distance
DBD3008	Feedrate velocity	REAL	if = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).
DBX3000.7	Start positioning axis	0|1	Reset does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	0|1	Interrupt signal, exits the function

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	Also 1, if override = 0 or setpoint position reached.
DBX3000.6	Position reached	1: Axis has reached the setpoint position.
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	



① First function activation using positive edge of *Start*.

② *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.

③ Positive acknowledgement *Position reached* = 1 and *Positioning axis active* = 1

④ Reset of function activation after receipt of acknowledgment

⑤ Signal change via function

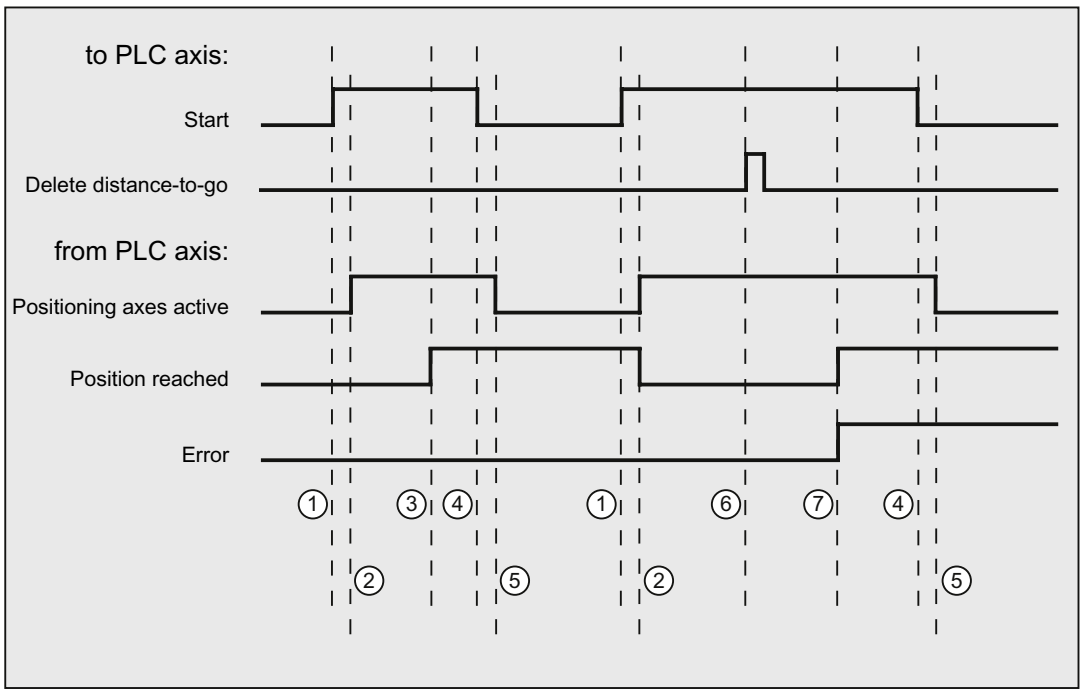
⑥ Positioning is interrupted by delete distance to go, signal duration min. 1 PLC cycle.

⑦ The signals *Position reached* and *Error* are reset, the *Error number* can be read (in this case, 30).

13.6.4.9 Positioning axis inch

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0|1	Only one of the bits must be set, if all bits are 0, then this means absolute positioning.
DBX3002.1	Shortest distance	0|1	
DBX3002.5	Direction of rotation as for M4	0	
DBX3003.0	Absolute, negative direction	0|1	
DBX3003.1	Absolute, positive direction	0|1	
DBD3002.2	Traversing dimension inch	1	Traversing dimension inch
DBD3002.3	Handwheel override	0	Override OFF
DBD3004	Setpoint position/setpoint distance	REAL	for "incremental": Setpoint distance
DBD3008	Feedrate velocity	REAL	if = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).
DBX3000.7	Start positioning axis	0|1	Reset does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	0|1	Interrupt signal, exits the function

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	Also 1, if override = 0 or setpoint position reached.
DBX3000.6	Position reached	1: Axis has reached the setpoint position.
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	



① First function activation using positive edge of *Start*.

② *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.

③ Positive acknowledgement *Position reached* = 1 and *Positioning axis active* = 1

④ Reset of function activation after receipt of acknowledgment

⑤ Signal change via function

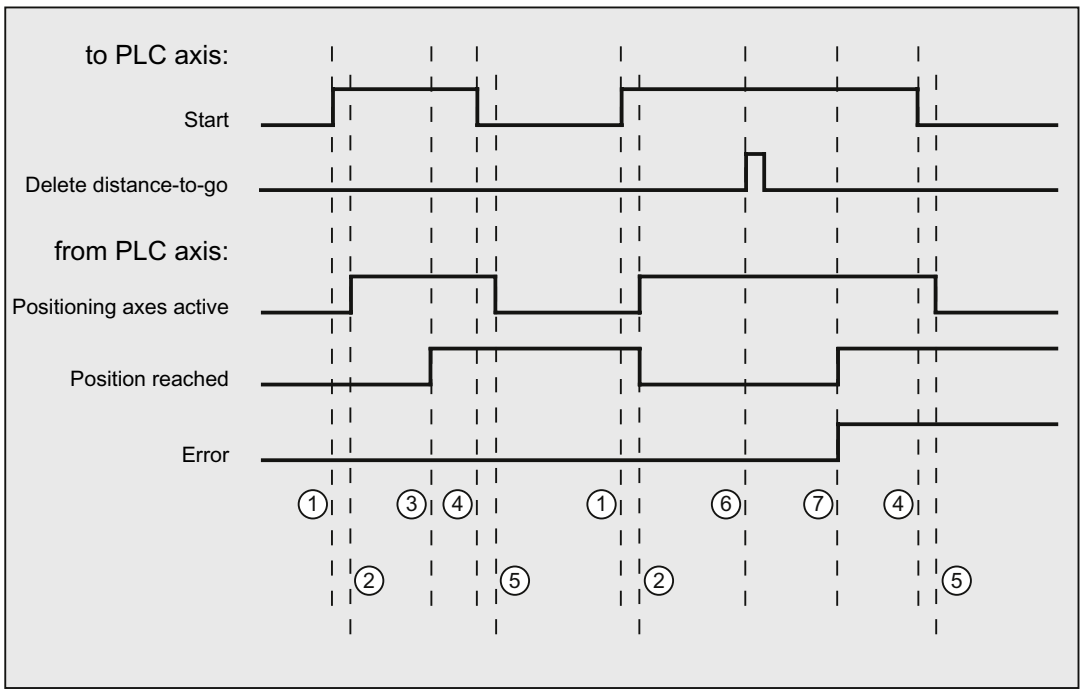
⑥ Positioning is interrupted by delete distance to go, signal duration min. 1 PLC cycle.

⑦ The signals *Position reached* and *Error* are reset, the *Error number* can be read (in this case, 30).

13.6.4.10 Positioning axis metric with handwheel override

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0|1	Only one of the bits must be set, if all bits are 0, then this means absolute positioning.
DBX3002.1	Shortest distance	0|1	
DBX3002.5	Direction of rotation as for M4	0	
DBX3003.0	Absolute, negative direction	0|1	
DBX3003.1	Absolute, positive direction	0|1	
DBD3002.2	Traversing dimension inch	0	Traversing dimension, metric
DBD3002.3	Handwheel override	1	Override ON
DBD3004	Setpoint position/setpoint distance	REAL	for "incremental": Setpoint distance
DBD3008	Feedrate velocity	REAL	if = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).
DBX3000.7	Start positioning axis	0|1	Reset does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	0|1	Interrupt signal, exits the function

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	Also 1, if override = 0 or setpoint position reached.
DBX3000.6	Position reached	1: Axis has reached the setpoint position.
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	



① First function activation using positive edge of *Start*.

② *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.

③ Positive acknowledgement *Position reached* = 1 and *Positioning axis active* = 1

④ Reset of function activation after receipt of acknowledgment

⑤ Signal change via function

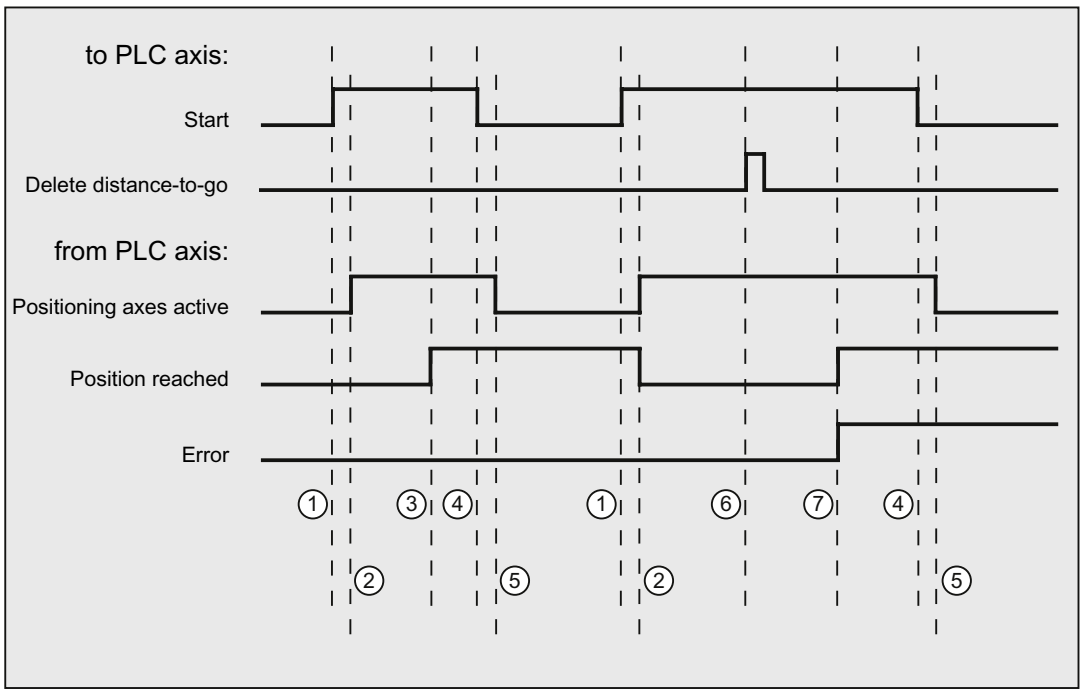
⑥ Positioning is interrupted by delete distance to go, signal duration min. 1 PLC cycle.

⑦ The signals *Position reached* and *Error* are reset, the *Error number* can be read (in this case, 30).

13.6.4.11 Positioning axis inch with handwheel override

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0|1	Only one of the bits must be set, if all bits are 0, then this means absolute positioning.
DBX3002.1	Shortest distance	0|1	
DBX3002.5	Direction of rotation as for M4	0	
DBX3003.0	Absolute, negative direction	0|1	
DBX3003.1	Absolute, positive direction	0|1	
DBD3002.2	Traversing dimension inch	1	Traversing dimension inch
DBD3002.3	Handwheel override	1	Override ON
DBD3004	Setpoint position/setpoint distance	REAL	for "incremental": Setpoint distance
DBD3008	Feedrate velocity	REAL	if = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).
DBX3000.7	Start positioning axis	0|1	Reset does not result in a stop!
DBX2.2	Delete distance-to-go, spindle reset	0|1	Interrupt signal, exits the function

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	Also 1, if override = 0 or setpoint position reached.
DBX3000.6	Position reached	1: Axis has reached the setpoint position.
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	



① First function activation using positive edge of *Start*.

② *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.

③ Positive acknowledgement *Position reached* = 1 and *Positioning axis active* = 1

④ Reset of function activation after receipt of acknowledgment

⑤ Signal change via function

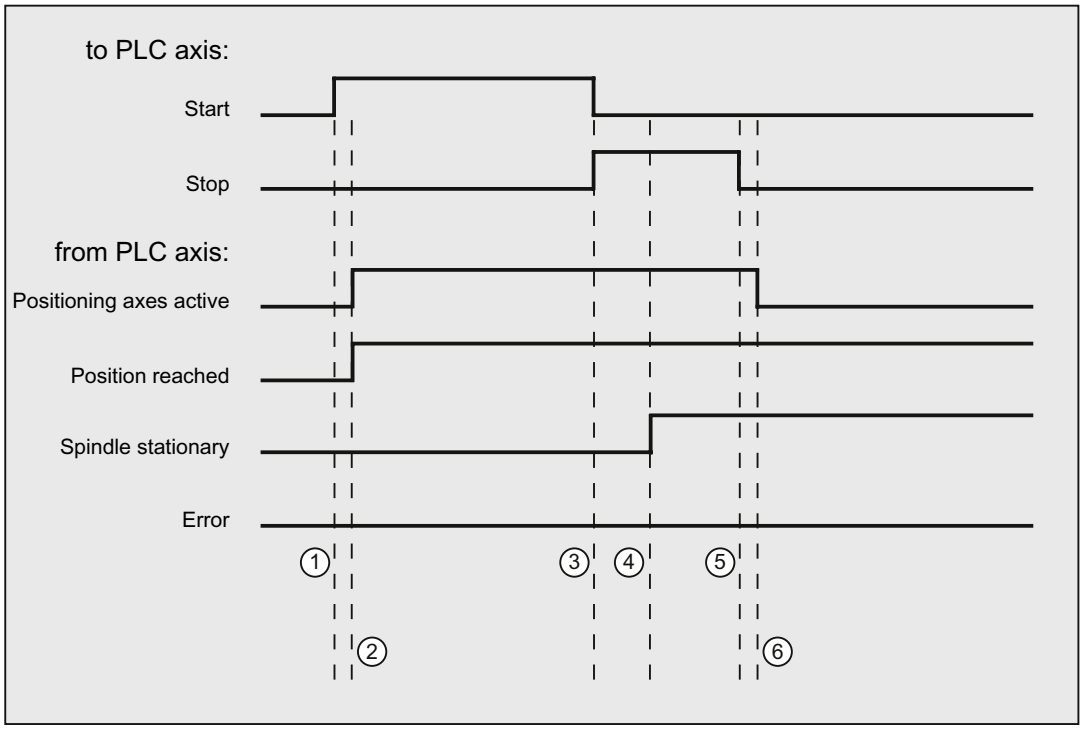
⑥ Positioning is interrupted by delete distance to go, signal duration min. 1 PLC cycle.

⑦ The signals *Position reached* and *Error* are reset, the *Error number* can be read (in this case, 30).

13.6.4.12 Rotate spindle with automatic gear stage selection

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	**0**	*Direction of rotation as for M4*:
DBX3002.1	Shortest distance	**0**	1: Direction of rotation specified by M4
DBX3002.5	Direction of rotation as for M4	**0|1**	0: Direction of rotation specified by M3
DBX3003.0	Absolute, negative direction	**0**	
DBX3003.1	Absolute, positive direction	**0**	
DBD3002.7	Automatic gear stage selection	**1**	Automatic gear stage selection ON
DBD3008	Feedrate velocity	**REAL**	Spindle speed
DBD3000.5	Start spindle rotation	**0|1**	
DBX3001.5	Stop spindle rotation	**0|1**	

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	1: for start or stop == 1,
DBX3000.6	Position reached	1: Setpoint speed is output
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	



① Function activated by user with a positive edge of *Start*.

② Signals *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.

③ The user stops spindle rotation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

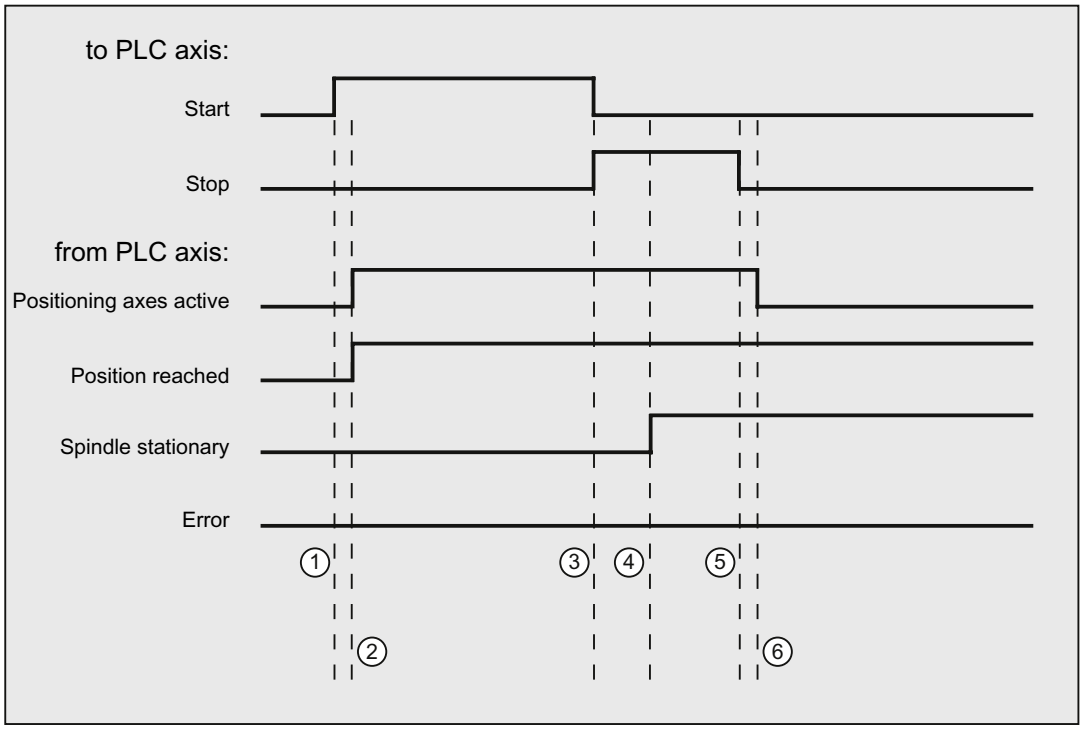
⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

13.6.4.13 Rotate spindle with constant cutting rate [m/min]

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0	*Direction of rotation as for M4*:
DBX3002.1	Shortest distance	0	1: Direction of rotation specified by M4
DBX3002.5	Direction of rotation as for M4	0|1	0: Direction of rotation specified by M3
DBX3003.0	Absolute, negative direction	0	
DBX3003.1	Absolute, positive direction	0	
DBD3002.2	Traversing dimension inch	0	Traversing dimension, metric
DBD3002.6	Const. Cutting rate	1	Constant cutting rate ON
DBD3008	Feedrate velocity	REAL	Spindle speed
DBD3000.5	Start spindle rotation	0|1	
DBX3001.5	Stop spindle rotation	0|1	

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	1: for start or stop == 1,
DBX3000.6	Position reached	1: Setpoint speed is output
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	



① Function activated by user with a positive edge of *Start*.

② Signals *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.

③ The user stops spindle rotation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

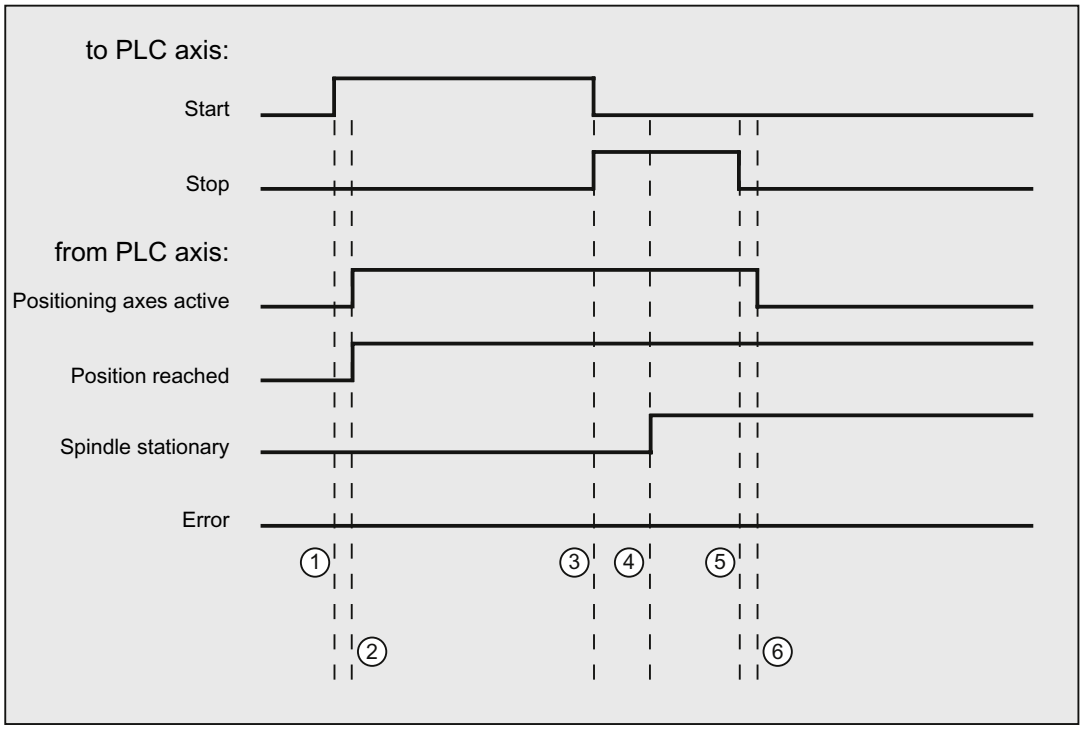
⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

13.6.4.14 Rotate spindle with constant cutting rate [feet/min]

DB380x	PLC → NCK control signals	Valid values	Remark
DBX3002.0	Incremental	0	*Direction of rotation as for M4*:
DBX3002.1	Shortest distance	0	1: Direction of rotation specified by M4
DBX3002.5	Direction of rotation as for M4	0|1	0: Direction of rotation specified by M3
DBX3003.0	Absolute, negative direction	0	
DBX3003.1	Absolute, positive direction	0	
DBD3002.2	Traversing dimension inch	1	Traversing dimension inch
DBD3002.6	Const. Cutting rate	1	Constant cutting rate ON
DBD3008	Feedrate velocity	REAL	Spindle speed
DBD3000.5	Start spindle rotation	0|1	
DBX3001.5	Stop spindle rotation	0|1	

DB390x	NCK → PLC status signals	Remark
DBX3000.7	Positioning axes active	1: for start or stop == 1
DBX3000.6	Position reached	1: Setpoint speed is output
DBX3000.1	Error	1: Error when traversing, evaluate error number in DBB3003!
DBB3003	Error number	
DBX1.4	Axis/spindle stationary	



① Function activated by user with a positive edge of *Start*.

② Signals *Positioning axis active* and *Position reached* are signaled back, *Position reached* is in this case irrelevant.

③ The user stops spindle rotation by resetting *Start* and setting *Stop*.

④ The spindle stops and the *Spindle stationary* signal is set.

⑤ The user then resets *Stop*.

⑥ Reset of *Stop* causes *Positioning axis active* to be reset.

13.6.4.15 Error messages

If a function could not be executed, this is indicated by the signal *Error* (DB390x .DBX3000.1 or DB390x.DBX3000.0) with a 'logical 1'. The cause of the error is coded as an error number:

Status		Meaning
Errors caused by PLC handling:		
1	16#01	Several functions of the axis/spindle were activated simultaneously
20	16#14	A function was started without the position being reached
30	16#1e	The axis/spindle was transferred to the NC while still in motion
40	16#28	The axis is programmed by the NC program, NCK internal error
50	16#32	Permanently assigned PLC axis: Traverses (JOG) or is referencing
60	16#3C	Permanently assigned PLC axis: Channel status does not permit a start
Errors that occur due to handling of the NCK: **The alarm numbers are described in the Diagnostics Manual Alarms SINUMERIK 828D, SINAMICS S120:**		
100	16#64	False position programmed for axis/spindle (corresponds to alarm number 16830)
101	16#65	Programmed speed is too high
102	16#66	Incorrect value range for constant cutting rate (corresponds to alarm number 14840)
104	16#68	Following spindle: Illegal programming (corresponds to alarm number 22030)
105	16#69	No measuring system available (corresponds to alarm number 16770)
106	16#6a	Positioning process of the axis still active (corresponds to alarm number 22052)
107	16#6b	Reference mark not found (corresponds to alarm number 22051)
108	16#6c	No transition from speed control to position control (corresponds to alarm number 22050)
109	16#6d	Reference mark not found (corresponds to alarm number 22051)
110	16#6e	Velocity/speed is negative
111	16#6f	Setpoint speed is zero
112	16#70	Invalid gear stage
115	16#73	Programmed position has not been reached
117	16#75	G96/G961 is not active in the NC
118	16#76	G96/G961 is still active in the NC
120	16#78	Axis is not an indexing axis (corresponds to alarm number 20072)
121	16#79	Indexing position error (corresponds to alarm number 17510)
125	16#7d	DC (shortest distance) not possible (corresponds to alarm number 16800)
126	16#7e	Absolute value minus not possible (corresponds to alarm number 16820)
127	16#7f	Absolute value plus not possible (corresponds to alarm number 16810)
128	16#80	No transverse axis available for diameter programming (corresponds to alarm number 16510)

Status		Meaning
130	16#82	Software limit switch plus (corresponds to alarm number 20070)
131	16#83	Software limit switch minus (corresponds to alarm number 20070)
132	16#84	Working area limit plus (corresponds to alarm number 20071)
133	16#85	Working area limit minus (corresponds to alarm number 20071)
134	16#85	Frame not permitted for indexing axis
135	16#87	Indexing axis with "Hirth-toothing" is active (corresponds to alarm number 17501)
136	16#88	Indexing axis with "Hirth toothing" is active and axis not referenced (corresponds to alarm number 17503)
137	16#89	Spindle operation not possible for transformed spindle/axis (corresponds to alarm number 22290)
138	16#8A	The corresponding effective coordinate-system-specific working area limit plus violated for the axis (corresponds to alarm number 20082)
139	16#8B	The corresponding effective coordinate-system-specific working area limit minus violated for the axis (corresponds to alarm number 20082)
System or other serious interrupts:		
200	16#c8	Corresponds to system alarm number 450007

13.6.5 Starting ASUBs

13.6.5.1 General

An ASUB (asynchronous subprogram) is an NC program that can be started by the PLC at any time, i.e. it is an NC interrupt program, because the running NC program is interrupted by the ASUB.

Requirement is that the ASUB is selected and parameterized using an NC program or using the PI service ASUB (Page 1029). By executing the PI service ASUB with the appropriate PI index, one of the two interrupts INT1 or INT2 is assigned the ASUB intended for the purpose.

Only one ASUB can be started at one time. If, in a PLC cycle, both start signals of the function interface described below are set to a logical 1, then the ASUBs are started in the sequence in which they are called. The user must set the start signal to a logical 0 if an acknowledgement was set in the interface for the job result.

Note

A start signal must not be set in the following cases:

- PI service "ASUB" has not been completed yet.
- DB2600.DBX0.1 == 1 (emergency stop)
- Request for a channel reset by the PLC is active.

  The ASUB can only be started again when the channel is in the "Reset" state AND DB3300.DBX3.7 == 1 (channel state reset).

13.6.5.2 Job start

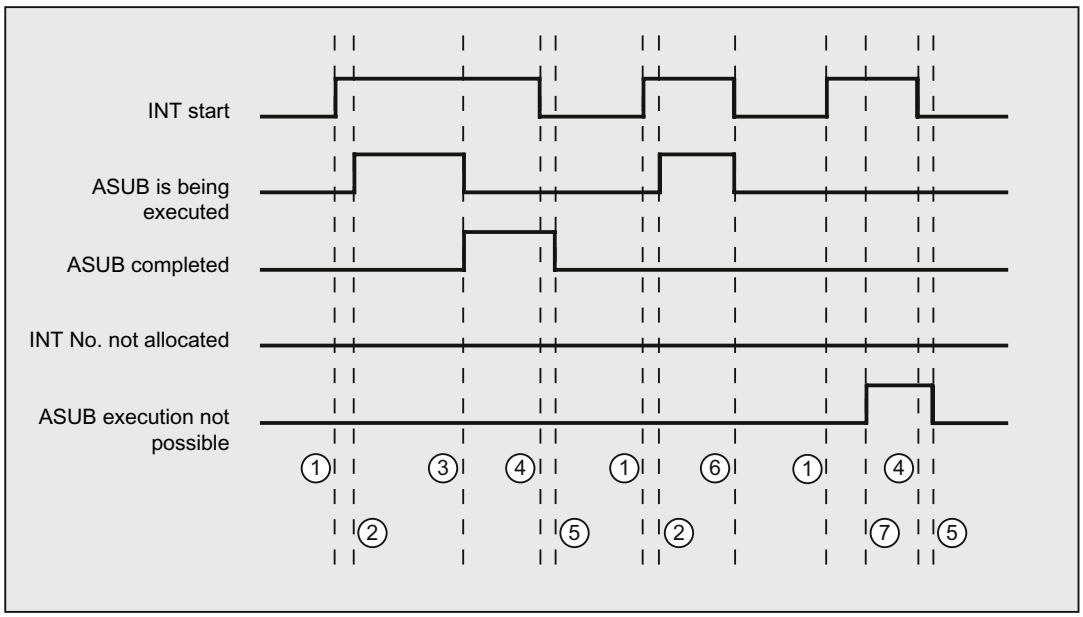
DB3400			ASUB: Job [r/w]					
			PLC -> NCK interface					
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBB 0000								INT1 start
DBB 0001								INT2 start
DBB 0002								
DBB 0003								
Start INT1, start INT2 Bit 0 = 1: Request to start the ASUB which is assigned to the relevant INT. Bit 0 = 0: End of the ASUB request after acknowledgement in the result interface DBB1000, DB1001.								

13.6.5.3 Job result

DB3400					ASUB: Result [r]			
					NCK -> PLC interface			
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBB 1000	INT1							
					ASUB cannot be executed 1)	Interrupt no. not assigned 2)	ASUB is being executed	ASUB completed 3)
DBB 1001	INT2							
					ASUB cannot be executed 1)	Interrupt no. not assigned 2)	ASUB is being executed	ASUB completed 3)
DBB 1002								
DBB 2003								
1) Negative acknowledgement: E.g. for emergency stop or channel reset request. 2) Negative acknowledgement: Number has not been assigned yet. Remedy: Execute PI service "ASUB". 3) Positive acknowledgement: ASUB successfully completed. Reset start signal (DB000, DB001).								

13.6.5.4 Signal flow

Signal flow



① Function activated by user with a positive edge of *Start*.
② *ASUB is being executed* is signaled back.
③ The acknowledgement *ASUB completed* indicates the successful execution, *ASUB is being executed* is withdrawn.
④ The signal to initiate the function is reset after receiving the acknowledgement from the user.
⑤ Signal change by the firmware.
⑥ Not permitted! If function activation is reset prior to receipt of acknowledgement, the output signals are not updated – without the operational sequence of the activated function being affected.
⑦ *ASUB cannot be executed*: Negative acknowledgement, error occurred.

Figure 13-7 Example: Signal flow

R1: Referencing 14

14.1 Brief Description

Function

The "Reference Point Approach" function is used to synchronize the measuring system of a machine axis with machine zero. The machine axis is traversed to machine zero and the measuring system set to zero.

If it is not possible to approach machine zero directly, a reference point within the traversing range of the machine axis is used whose position with reference to machine zero precisely known.

After the reference point approach, the measuring system of the machine axis is not set to zero but to the corresponding reference point value.

Measuring systems and referencing methods

The "Reference point approach" function enables machine axes to be referenced using the following measuring systems and referencing methods:

- Measuring systems
  - Incremental rotary measuring system with at least one zero mark
  - Incremental linear measuring system
  - Rotary measuring system with distancecoded reference marks (supplied by Heidenhain)
  - Linear measuring system with distancecoded reference marks (supplied by Heidenhain)
  - Absolute rotary measuring system
  - Absolute linear measuring system
- Referencing methods
  - Referencing with incremental measuring systems with BERO and one-edge and two-edge detection
  - Referencing with incremental measuring systems with replacement of homing cam with BERO
  - Referencing with incremental measuring systems with BERO with configured approach velocity for spindle applications

- Referencing with measuring systems with distancecoded reference marks by overtravelling 2 or 4 zero marks
- Referencing of passive measuring systems using measuring system adjustment
- Referencing in followup mode
- Referencing with cam switch at the drive

Start

The reference point approach of a machine axis can be started manually or via the part program:

- Manual: Operation mode JOG and MDA, machine function REF
- Part program: Part program command `G74`

14.2 Axisspecific referencing

In axis-specific reference point approach, reference point approach must be initiated individually for each machine axis that is to be referenced.

Selecting mode and machine function

Before starting reference point approach of the machine axes, you must first place the relevant mode group in JOG or MDA mode:

DB11, ... DBX0.2 (active JOG mode)

DB11, ... DBX0.1 (active JOG mode)

Then machine function REF (reference point approach) must be selected:

DB11, ... DBX1.2 (REF machine function)

Start of reference point approach

In axis-specific reference point approach, each machine axis must be started individually.

Reference point approach is started with the axis-specific traversing keys:

DB31, ... DBX4.6 (Traversing key minus)

DB31, ... DBX4.7 (Traversing key minus)

Direction enable

To avoid faulty operation, the direction release must be parameterized:

MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point in minus direction)

The direction enable specifies which traversing key starts the reference point approach:

Value	Description
0	Reference point approach in plus direction
1	Reference point approach in minus direction

Jog mode

The following machine data element can be used to specify whether reference point approach is completed when the direction key is pressed once or whether the operator is required to keep the direction key pressed (jogging) for safety reasons:

MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD (INC and REF in jog mode)

If the machine operator releases the direction key, the machine axis is decelerated to zero speed. Reference point approach is not aborted. Reference point approach is continued the next time the direction key is pressed.

Referencing status

The referencing status of the machine axis is reset with the start of the reference point approach:

DB31, ... DBX60.4 (referenced / synchronized 1)

DB31, ... DBX60.5 (referenced / synchronized 2)

DB21, ... DBX36.2 (all axes with obligatory reference point are referenced)

Distance-coded measuring systems

In distance-coded measuring systems, reference point approach can be started with any traversing key.

Sequence

The machine operator or machine manufacturer (via the PLC user program) is responsible for ensuring that the machine axes are referenced in the proper order.

- Machine operator

  The machine axes must be started by the machine operator in the specified order.

- Machine manufacturer

  The PLC user program of the machine manufacturer allows machine axes to be started only in the proper order.

Simultaneous reference point approach of several machine axes

Several machine axes can be referenced simultaneously, depending on the control:

SINUMERIK 840D:	max. 8 machine axes

Completion of reference point approach

Acknowledgment that reference point approach of a machine axis has been successfully completed is given by setting the referencing status:

DB31, ... DBX60.4 (referenced / synchronized 1)

DB31, ... DBX60.5 (referenced / synchronized 2)

Cancellation of reference point approach

In axis-specific reference point approach, the machine axis is traversed in the channel that was assigned as the master channel of the machine axis.

MD30550 $MA_ AXCONF_ASSIGN_MASTER_CHAN

For aborting the reference point approach, either mode group reset or channel reset for the master channel of the machine axis must be activated:

DB11, ... DBX0.7 (mode group reset)

DB21, ... DBX7.7 (channel reset)

All machine axes that have not yet successfully completed reference point approach when the action is cancelled remain in status "Not referenced":

DB31, ... DBX60.4 (referenced / synchronized 1)

DB31, ... DBX60.5 (referenced / synchronized 2)

14.3 Channelspecific referencing

In channel-specific reference point approach, all machine axes of the channel are referenced in the parameterized sequence when reference point appraoch is initiated.

Selecting mode and machine function

Before starting reference point approach of the machine axes, you must first set the mode group to JOG or MDA mode:

DB11, ... DBX0.2 (active JOG mode)

DB11, ... DBX0.1 (active MDA mode)

Then machine function REF (reference point approach) must be selected:

DB11, ... DBX1.2 (REF machine function)

Parameterizing the axis sequence

The following machine data element is used to specify the sequence in which the machine axes of the channel are referenced:

MD34110 $MA_REFP_CYCLE_NR = *Number*

Number	Description
-1	The machine axis does not have to be referenced for NC START in the channel.
0	The machine axis does not participate in channel-specific reference point approach.
1 - 15	Sequence number in channel-specific reference point approach.

The machine axes are referenced in ascending order of numbers.

Machine axes with the same number will be referenced simultaneously.

Simultaneous reference point approach of several machine axes

Several machine axes can be referenced simultaneously, depending on the control:

SINUMERIK 840D:	Max. 8 machine axes
SINUMERIK 810D:	Max. 5 machine axes

Start of reference point approach

Channel-specific reference point approach is started with:

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

The status of channel-specific reference point approach is indicated by the channel with:

DB21, ... DBX33.0 (activate referencing)

Referencing status

The referencing status of the machine axis is reset with the start of the reference point approach:

DB31, ... DBX60.4 (referenced / synchronized 1)

DB31, ... DBX60.5 (referenced / synchronized 2)

Completion of reference point approach

As soon as channel-specific reference approach has been successfully completed for all machine axes involved, this is acknowleged with:

DB21, ... DBX36.2 (all axes with obligatory reference point are referenced)

Cancellation of reference point approach

In channel-specific reference point approach the machine axis is traversed in the channel to which that axis is currently assigned as channel axis.

For aborting the reference point approach either mode group reset or channel reset for the corresponding channel must be activated:

DB11, ... DBX0.7 (mode group reset)

DB21, ... DBX7.7 (channel reset)

All machine axes for which the reference point approach is not yet successfully completed when the action is cancelled remain in status "Not referenced":

DB31, ... DBX60.4 (referenced / synchronized 1)

DB31, ... DBX60.5 (referenced / synchronized 2)

14.4 Reference point appraoch from part program (G74)

Referencing of machine axes can be activated for the first time or repeated from the part program

Referencing must be repeated, for example, after:

- converting the actual value of the machine axis: PRESETON function
- Machine axis is parked:

  DB31, ... DBX1.5 (position measuring system 1) = 0

  DB31, ... DBX1.6 (position measuring system 2) = 0
- DB31, ... DBX2.1 (servo enable) = 0
- Exceeding the encoder limit frequency of the position measuring system

Programming

Syntax

G74*Machine axis* { *Machine axis* }

Function

Machine axes can be referenced from a part program with part program instruction G74

Parameter: *Machine axes*

The name of the machine axis must be specified. The machine axis must be a channel axis of the channel in which the part program is processed.

Effective:

G74 is non-modal.

Special features

G74 must be programmed in a separate part program block.

Reset response

Mode group reset or channel reset aborts the reference point approach for all programmed machine axes:

DB11, ... DBX0.7 (mode group reset)

DB21, ... DBX7.7 (channel reset)

All machine axes for which the reference point approach is not yet successfully completed when the action is cancelled remain in status "Not referenced":

DB31, ... DBX60.4 (referenced / synchronized 1)

DB31, ... DBX60.5 (referenced / synchronized 2)

14.5 Referencing with incremental measurement systems

14.5.1 Hardware signals

Depending on the machine design and the properties of the incremental measuring system used, different hardware signals must be connected.

Reference cam

- Connection
  The reference cam signal can be connected to a digital input of an external PLC I/O module or to a fast input on the NCU X142 interface.
- NC/PLC interface signal
  The reference cam signal must be transferred from the PLC user program to the axial NC/PLC interface:
  DB31, ... DBX12.7 (deceleration of reference point approach)

Zero mark selection

If during the reference point approach of the axis or spindle several zero marks of the measuring system are detected (e.g. measuring gear between the motor and encoder), then the specific zero mark must be selected with an additional BERO signal.

- Connection
  The BERO must be connected to a fast digital input on the NCU X122 or X132 interface.
- Activation
  In order that the BERO signal is evaluated, the digital input to which the BERO is connected must be selected in drive parameter p0493 for the axis/spindle.

Equivalent zero mark

If the used measuring system does not provide a zero mark signal, an equivalent zero mark can be created via a BERO signal.

- Connection
  The BERO must be connected to a fast digital input on the NCU X122 or X132 interface.
- Activation
  In order that the BERO signal is evaluated, the digital input to which the BERO is connected must be selected in drive parameter p0494 or p0495 for the axis/spindle.

Overview

Signal	Connection: Dig. input via	Set
Reference cam	Ext. PLC I/O module or NCU: X142	PLC user program: DB31, ... DBX12.7
Zero mark selection	NCU: X122 or X132	Drive parameter: p0493
External zero mark or equivalent zero mark	NCU: X122 or X132	Drive parameter: p0494 or p0495

References

- NCU interfaces: SINUMERIK 840D sl Manual, NCU7x0.3 PN, Section "Connecting" > "Digital I/Os"
- Drive parameters: SINAMICS S120/S150 List Manual

14.5.2 Zero mark selection with BERO

Function

Referencing of incremental measuring systems is based on the unique position of the encoder zero mark relative to the overall traversing range of the machine axis. If because of machine-specific conditions, several encoder zero marks are detected in the traversing range of the machine axis (for examples, see figure below), a BERO must be mounted on the machine for clear determination of the reference point. The position of the reference point is then derived from the combination of BERO signal and encoder zero mark.

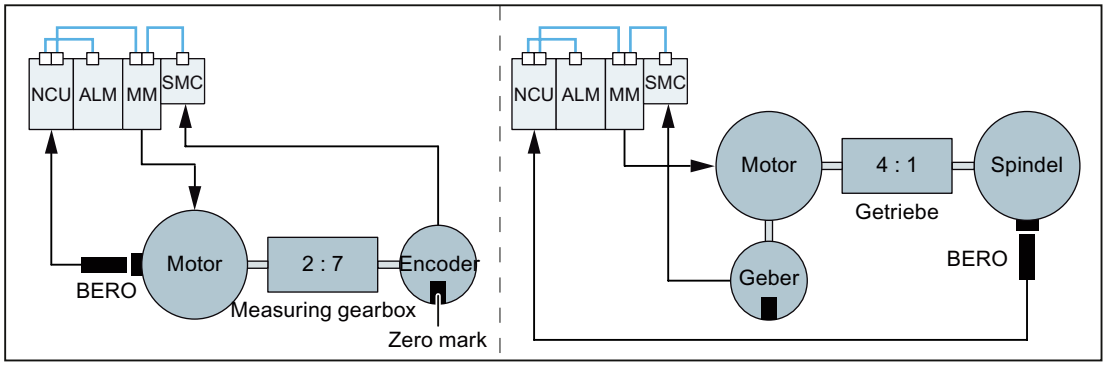


Figure 14-1 Measuring gear between the motor and encoder or reduction gear between the motor and spindle

Parameterization

NC: Referencing mode

"Referencing of incremental, rotary or linear measuring systems: Zero pulse on the encoder track" should be parameterized as referencing mode:

MD34200 $MA_ENC_REFP_MODE[<axis>] = 1

Drive: Zero mark selection

The digital input on the NCU interface to which the BERO is connected must be set in parameter p0493.

Note

BERO signal: Zero mark selection

The processing of the BERO signal is performed exclusively in the drive. Connection and parameterization, see Section "Hardware signals (Page 1327)".

14.5.3 Time sequence

Reference point approach with incremental measuring systems can be divided into three phases:

- Phase 1: "Phase 1: Traversing to the reference cam (Page 1331)"
- Phase 2: "Phase 2: Synchronization with the zero mark (Page 1334)"
- Phase 3: "Phase 3: Traversing to the reference point (Page 1339)"

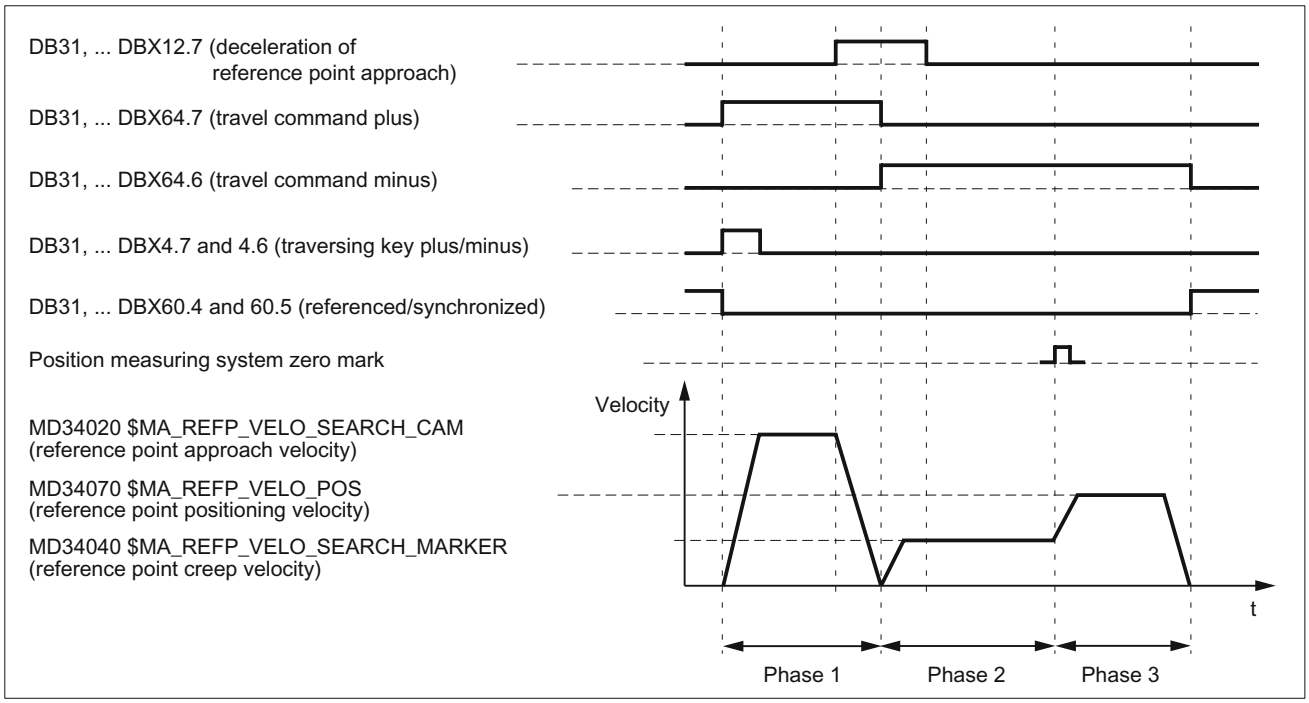


Figure 14-2 Time sequence when referencing with incremental measuring systems (example)

14.5.4 Phase 1: Traversing to the reference cam

Phase 1: Graphic representation

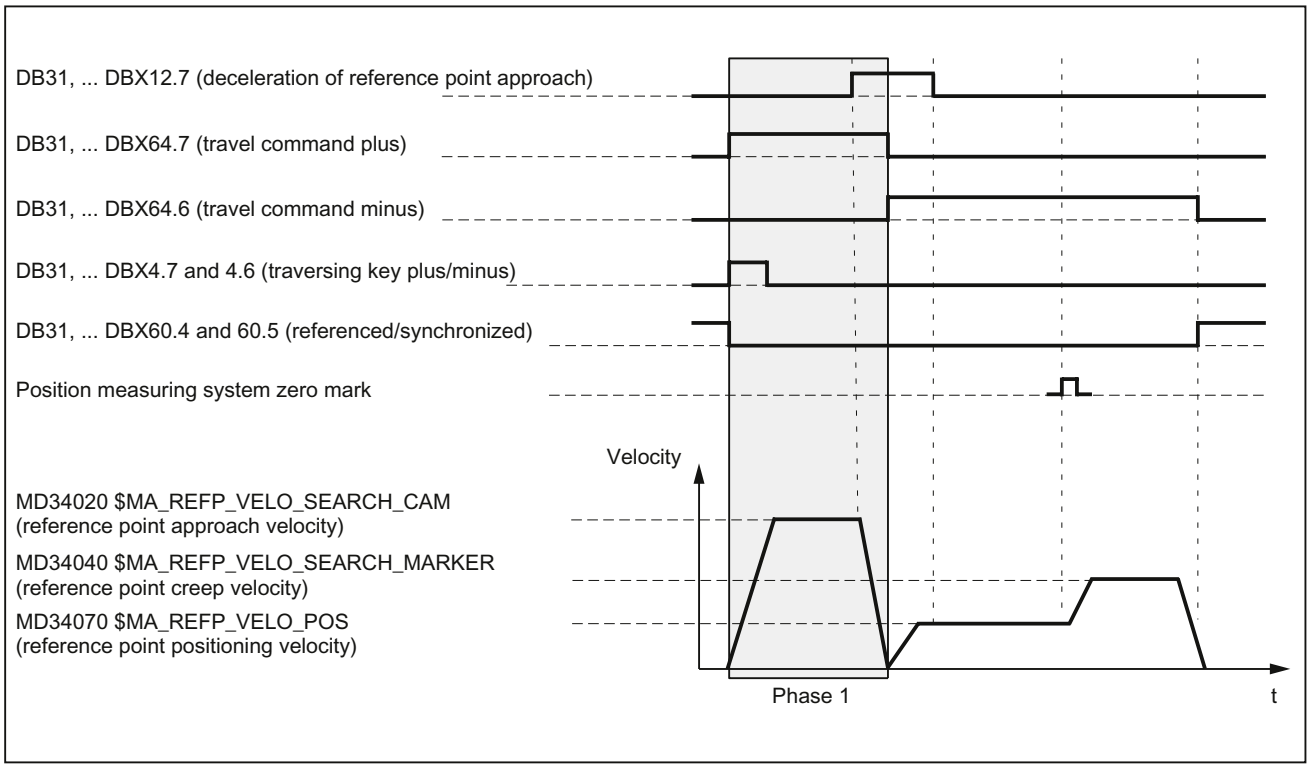


Figure 14-3 Phase 1: Traversing to the reference cam

Phase 1: Start

To start the reference point approach, see Sections "Axisspecific referencing (Page 1321)" and "Axisspecific referencing (Page 1321)".

Phase 1: Sequence

In Phase 1, depending on the position of the machine axis with reference to the reference cam, we distinguish between three cases:

1. The machine axis is positioned before the reference cam
2. The machine axis is positioned on the reference cam
3. The machine axis has no reference cam

Case 1: The machine axis is positioned before the reference cam

After the start of reference point approach, the machine axis is accelerated in the parameterized direction and to the parameterized reference point approach velocity :

- MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point in minus direction)
- MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach velocity)

The reaching of the reference cam must detected by querying a digital input in the PLC user program and communicated to the NC via the following interface signal:

DB31, ... DBX12.7 = 1 (reference point approach deceleration)

With detection of the NC/PLC interface signal, the machine axis is decelerated to zero speed. Whereby at least the distance s_{min} is traversed. This ensures that the machine axis leaves the reference cam in Phase 2 with the parameterized reference point creep velocity.

$$SMIN = \frac{(MD34040\ \$MA\_REFP\_VELO\_SEARCH\_MARKER)2}{2 * MD32300\ \$MA\_MAX\_AX\_ACCEL}$$

Phase 1 is now complete. Reference point approach is continued with Phase 2.

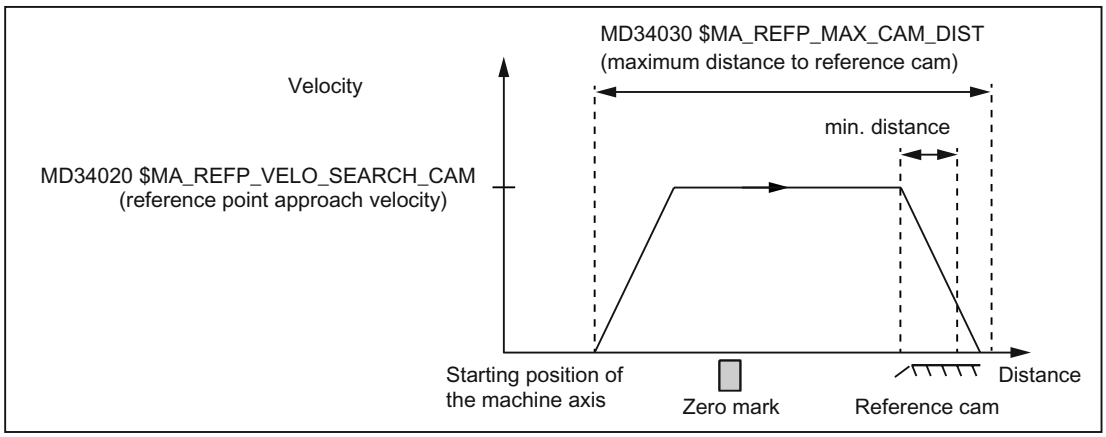


Figure 14-4 Minimum distance for deceleration

Case 2: The machine axis is positioned on the reference cam

The machine axis remains at its starting position.
Phase 1 is now complete. Reference point approach is continued with Phase 2.

Case 3: The machine axis has no reference cam

Machine axes without reference cams remain at their starting position.

These include, for example:

- Machine axes that only have one zero mark along their entire traversing range
- Rotary axes that only have one zero mark per revolution

Zero must be entered in the following machine data for machine axes without a reference cam:

MD34000 $MA_REFP_CAM_IS_ACTIVE = 0 (axis with reference cam)

Phase 1 is now complete. Reference point approach is continued with Phase 2.

Phase 1: Properties

- Feed override active.
- Feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.
- The machine axis is stopped if the reference cam is not reached within the parameterized maximum distance:

  MD34030 $MA_REFP_MAX_CAM_DIST (max. distance to the reference cam)

See also

Channelspecific referencing (Page 1324)

14.5.5 Phase 2: Synchronization with the zero mark

Phase 2: Graphic representation

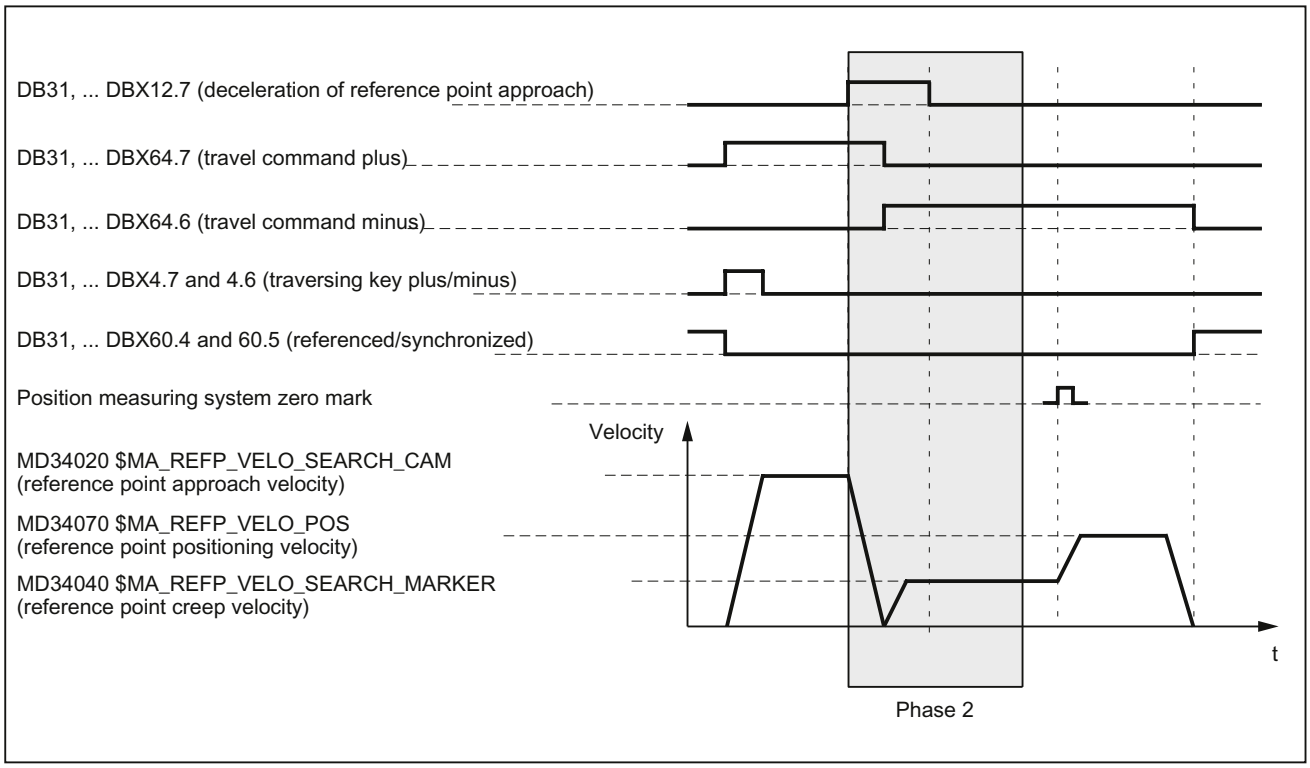


Figure 14-5 Phase 2: Synchronization with the zero mark

Phase 2: Start

Phase 2 is automatically started when Phase 1 has been completed without an alarm.

Initial situation:

The machine axis is positioned on the reference cam.

Zero mark search direction:

The direction of the zero mark search results from the settings in the machine data:

- MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point in minus direction)
- MD34050 $MA_REFP_SEARCH_MARKER_REVERSE (direction reversal on reference cam)

Phase 2: Sequence

The synchronization in Phase 2 can be performed via the falling or rising edge of the reference cam. The parameterization is performed via:

MD34050 $MA_REFP_SEARCH_MARKER_REVERSE[<axis>] = <value>

Value	Meaning
0	Synchronization with falling reference cam edge
1	Synchronization with rising reference cam edge

Note

If the actual velocity of the machine axis at approach of the reference cam has not yet reached the target velocity of Phase 2 within the parameterized tolerance limits, Phase 1 will be restarted. This will be the case, for example, if the machine axis is positioned on the reference cam when reference point approach is started.

MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance)

Case 1: Synchronization with falling reference cam edge

During synchronization with falling reference cam edge, the machine axis accelerates to the parameterized reference point creep velocity opposite to the parameterized reference point approach direction (traversing direction of Phase 1)

After leaving the reference cam, the machine axis waits for the next encoder zero mark: DB31, ... DBX12.7 == 0

As soon as the encoder zero mark is detected, Phase 2 comes to an end. The machine axis continues at constant velocity and reference point approach is continued with Phase 3.

- MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

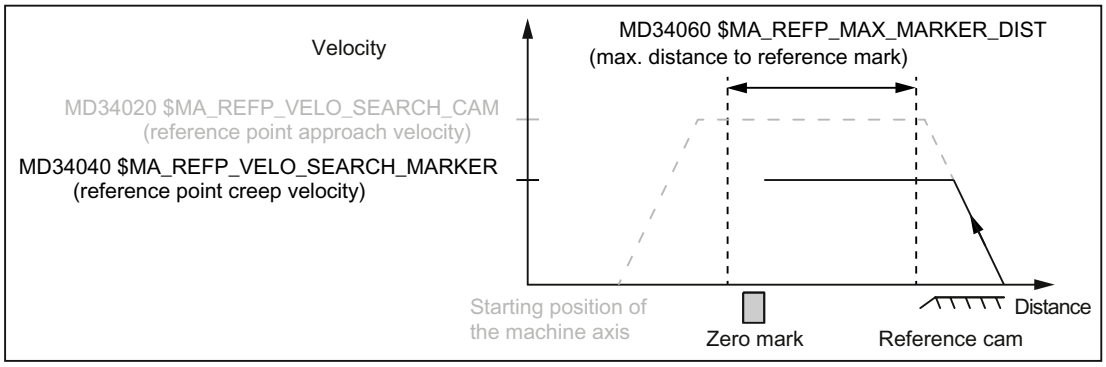


Figure 14-6 Synchronization with falling reference cam edge

Case 2: Synchronization with rising reference cam edge

During synchronization with rising reference cam signal edge, the machine axis accelerates to the parameterized reference point approach velocity against the parameterized reference point approach direction (traversing direction of the Phase 1):

- MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach velocity)
- MD34010 $MA_REFP_CAM_DIR_IS_MINUS (reference point approach in minus direction)

After leaving the reference cam, the machine axis decelerated to standstill:
DB31, ... DBX12.7 == 0

The machine axis then travels back to the reference cam at the parameterized reference point creep velocity:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

After reaching the reference cam (DB31, ... DBX12.7 = 1), the machine axis waits for the next encoder zero mark.

As soon as the encoder zero mark is detected, Phase 2 comes to an end. The machine axis continues at constant velocity and reference point approach is continued with Phase 3.

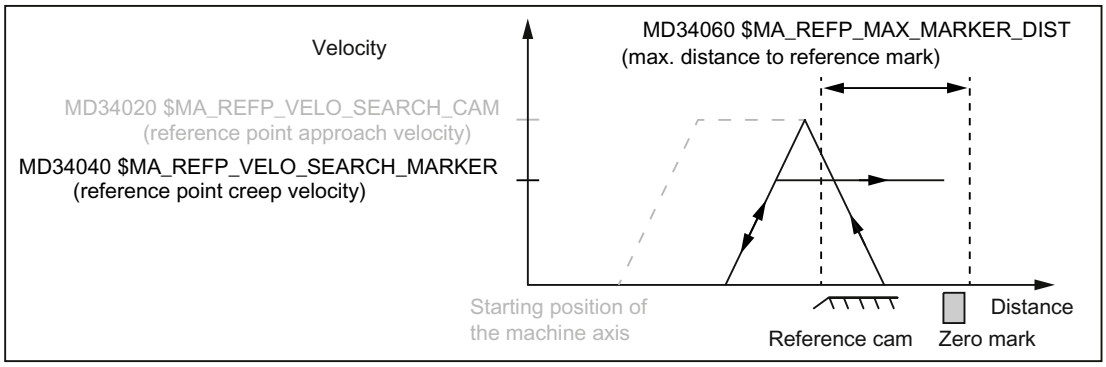


Figure 14-7 Synchronization with rising reference cam edge

Electronic reference cam shift

The electronic reference cam shift is used to compensate for expansions of the reference cam caused by temperature so that synchronization is always to the same encoder zero mark:

MD34092 $MA_ REFP_CAM_SHIFT (electronic reference cam shift for incremental measuring systems with equidistant zero marks)

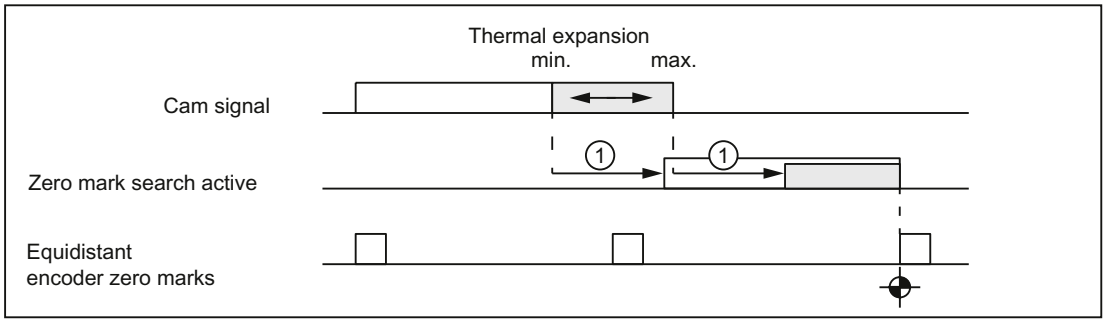
With the electronic reference cam shift, synchronization is not performed immediately to the next encoder zero mark after detection of the reference cam edge, but only after the parameterized offset distance has been traversed.

Due to the determination of the distance traversed in the interpolation cycle since the detection of the reference cam edge, the effective shift distance is s_{shift}:

s_{shift_min} = MD34092 $MA_ REFP_CAM_SHIFT

s_{shift_max} = MD34092 $MA_ REFP_CAM_SHIFT +
MD34040 $MA_REFP_VELO_SEARCH_MARKER * interpolation cycle

The electronic reference cam shift acts in the direction of zero mark search.



① Reference cam shift

Figure 14-8 Electronic reference cam shift

Requirement

The electronic reference cam shift is only active for machine axes with reference cam:

MD34000 $MA_REFP_CAM_IS_ACTIVE == 1

Reference cam adjustment

Encoder with equidistant zero marks

Always ensure that the reference cam of encoders that supply zero marks at equidistances is accurately adjusted so that the correct zero mark is always detected during reference point approach.

Dynamic response

The following factors influence the dynamic response from the arrival of the reference cam to the machine up to the detection of reference cam signals transferred from the PLC user program to the NC:

- Switching accuracy of the reference cam switch
- Delay of the reference cam switch (NC contact)
- Delay at the PLC input
- PLC cycle time
- Cycle time for updating the NC/PLC interface
- Interpolation cycle
- Position control cycle

Notes on setting

- Reference cam

  Aligning the signal edge of the reference cam directly between two zero marks has proven to be the most practical method.

- Electronic reference cam shift

  Information needed for parameterizing the electronic reference cam shift is to be found in the read-only machine data:

  MD34093 $MA_REFP_CAM_MARKER_DIST (distance between reference cam/reference mark)

  The indicated value is equivalent to the distance between departure from the reference cam and detection of the reference mark. If the values are too small, there is a risk that the determination of the reference point will be non-deterministic, due to temperature effects or fluctuations in the operating time of the cam signal.

⚠ WARNING

If the reference cam adjustment is faulty or inaccurate, an incorrect zero mark can be evaluated. The controller then calculates an incorrect machine zero. As a result, the machine axis will approach the wrong positions. Software limit switches, protected areas and working area limitations act on incorrect positions and are therefore incapable of protecting the machine. The path difference is +/- of the path covered by the machine axis between two zero marks.

Phase 2: Properties

- Feedrate override is **not** active.

  Machine axis moves internally when feedrate override = 100%.

  If a feedrate override of 0% is specified, an abort occurs.
- Feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are **not** active.
- If the machine axis does not arrive at Phase 2 within the parameterized distance of the reference mark (encoder zero mark), the machine axis will be stopped:

  MD34060 $MA_REFP_MAX_ MARKER_DIST (max. distance to the reference mark)

14.5.6 Phase 3: Traversing to the reference point

Phase 3: Graphic representation

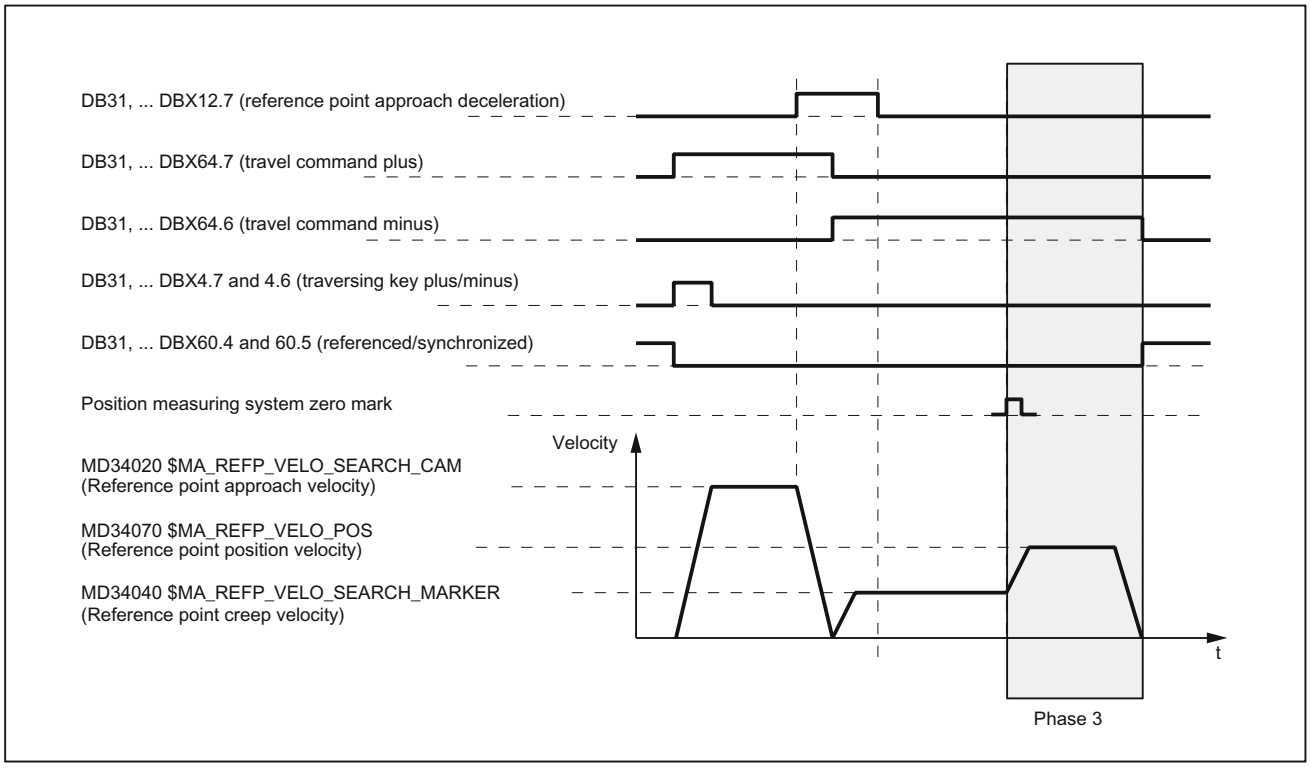


Figure 14-9 Phase 3: Traversing to the reference point

Phase 3: Start

At the end of Phase 2 the machine axis travels at reference point creep velocity. Therefore, as soon as Phase 2 is completed successfully without an alarm, Phase 3 is started without interruption.

Initial situation

The encoder zero mark has been detected.

Phase 3: Sequence

The machine axis moves at the assigned reference point positioning velocity:
MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)
from the encoder zero mark detected in Phase 2 to the reference point.

The path s_{ref} to be covered is calculated from the sum of the reference point distance plus reference point offset:

MD34080 $MA_REFP_MOVE_DIST (reference point distance)

MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)

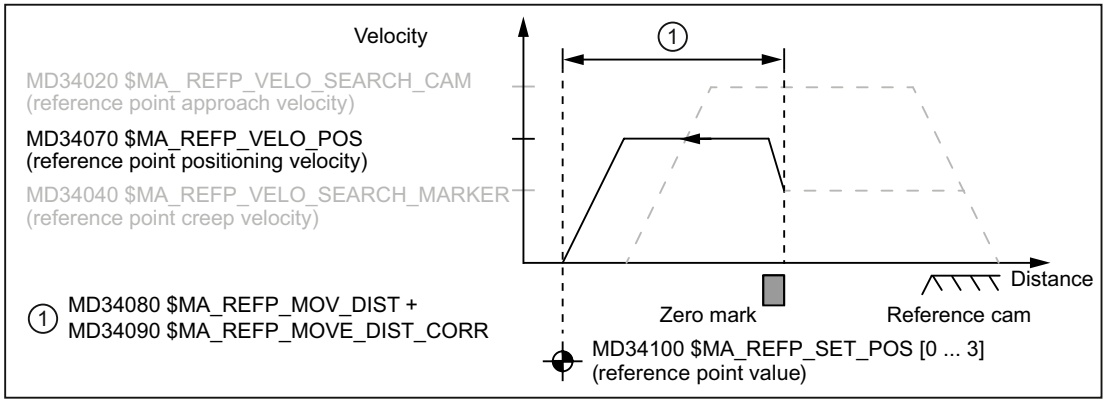


Figure 14-10 Reference point position

When the reference point is reached, the machine axis is stopped and the actual-value system is synchronized with the reference point value n specified by the NC/PLC interface.

MD34100 $MA_ REFP_SET_POS[<n>] (reference point value)

The selection of the reference point value is performed via the NC/PLC interface:

DB31, ... DBX2.4 ... 7 (reference point value 1 ... 4)

The actual-value system is synchronized to the reference point value that was selected at the time the reference cam was reached in Phase 1 (DB31, ... DBX12.7 == 1).

The machine axis is now referenced. The interface signal is set as feedback to the PLC user program, depending on the active measuring system:

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

Features of Phase 3

- Feed override active.
- Feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.

Special feature of Phase 3

In the following cases, the machine axis stops first after detection of the zero mark and then traverses back to the reference point:

- Because of the reference point positioning velocity, the sum of reference point distance and reference point offset is less than the required braking distance:
  MD34080 + MD34090 < "required braking distance due to MD34070"
- The reference point is located, opposite to the current travel direction, "behind" the reference cam.

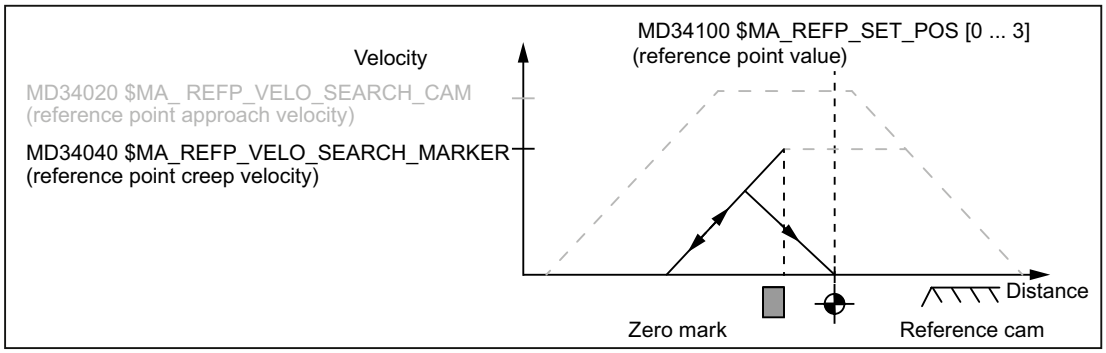


Figure 14-11 Reference point distance plus reference point offset less than braking distance

14.6 Referencing with distance-coded reference marks

14.6.1 General overview

Distancecoded reference marks

Measuring systems with distance-coded reference marks consist of two parallel scale tracks:

- Incremental grating
- Reference mark track

The distance between any two consecutive reference marks is defined. This makes it possible to determine the absolute position of the machine axis when two consecutive reference marks are crossed. For example, if the distance between the reference marks is approx. 10 mm, a traverse path of approx. 20 mm is all that is required to reference the machine axis.

Referencing can be performed from any axis position in the positive or negative direction (exception: end of travel range).

14.6.2 Basic parameter assignment

Linear measuring systems.

The following data must be set to parameterize linear measuring systems:

- The absolute offset between the machine zero point and the position of the first reference mark of the linear measuring system:

  MD34090 $MA_REFP_MOVE_DIST_CORR (reference point/absolute offset)

  See also below: Determining the absolute offset

- Orientation of the length measuring system (equidirectional or inverse) relative to the machine system coordinate system:

  MD34320 $MA_ENC_INVERS (length measuring system inverse to the machine system)

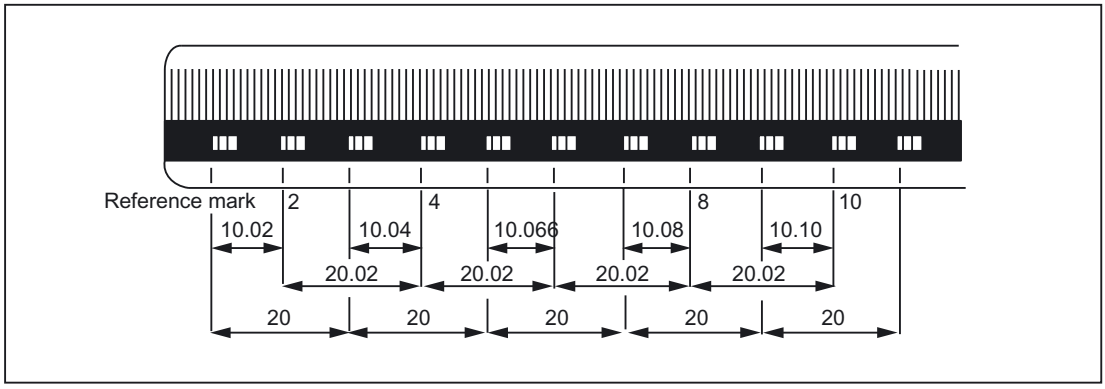


Figure 14-12 DIADUR graduated glass scale with distance-coded reference marks (dimensions in mm for 20 mm scale division)

Rotary measuring system

For rotary measuring systems, the same applies as for linear measuring systems (see above).

Determining the absolute offset

The following procedure is recommended for the determination of the absolute offset between the machine zero point and the position of the first reference mark of a machine axis:

1. Enter the value zero for the absolute offset:

   MD34090 $MA_REFP_MOVE_DIST_CORR = 0

2. Perform reference point approach.

   Note: Reference point approach should be performed at a point in the machine where the exact position of the machine axis relative to machine zero can be determined easily with a laser interferometer, for example.

3. Determine the actual position of the machine axis via the operator interface screen.
4. Measure the current position of the machine axis with reference to the machine zero point.
5. Calculate absolute offset and enter in MD34090.

The absolute offset is calculated with respect to the machine coordinate system and depending on the orientation of the measuring system (equidirectional or inverse) as:

Orientation of the measuring system	Absolute offset
equidirectional	Measured position + displayed actual position
Opposite direction	Measured position - displayed actual position

⚠WARNING

After determining the absolute offset and the entry in MD34090, the reference point traversing for the machine axis must be carried out once more.

Referencing methods

Referencing with distance-coded reference marks can be performed in one of two ways:

- Evaluation of **two** consecutive reference marks:

  MD34200 $MA_ENC_REFP_MODE (referencing mode) = 3

  Advantage:

  – Short travel path

- Evaluation of **four** consecutive reference marks:

  MD34200 $MA_ENC_REFP_MODE = 8

  Advantage:

  – Plausibility check by NC is possible

  – Increase in reliability of referencing result

14.6.3 Time sequence

Time sequence

Referencing with distance-coded reference marks can be divided into two phases:

- Phase 1: Travel across the reference marks with synchronization
- Phase 2: Travel to a fixed destination point

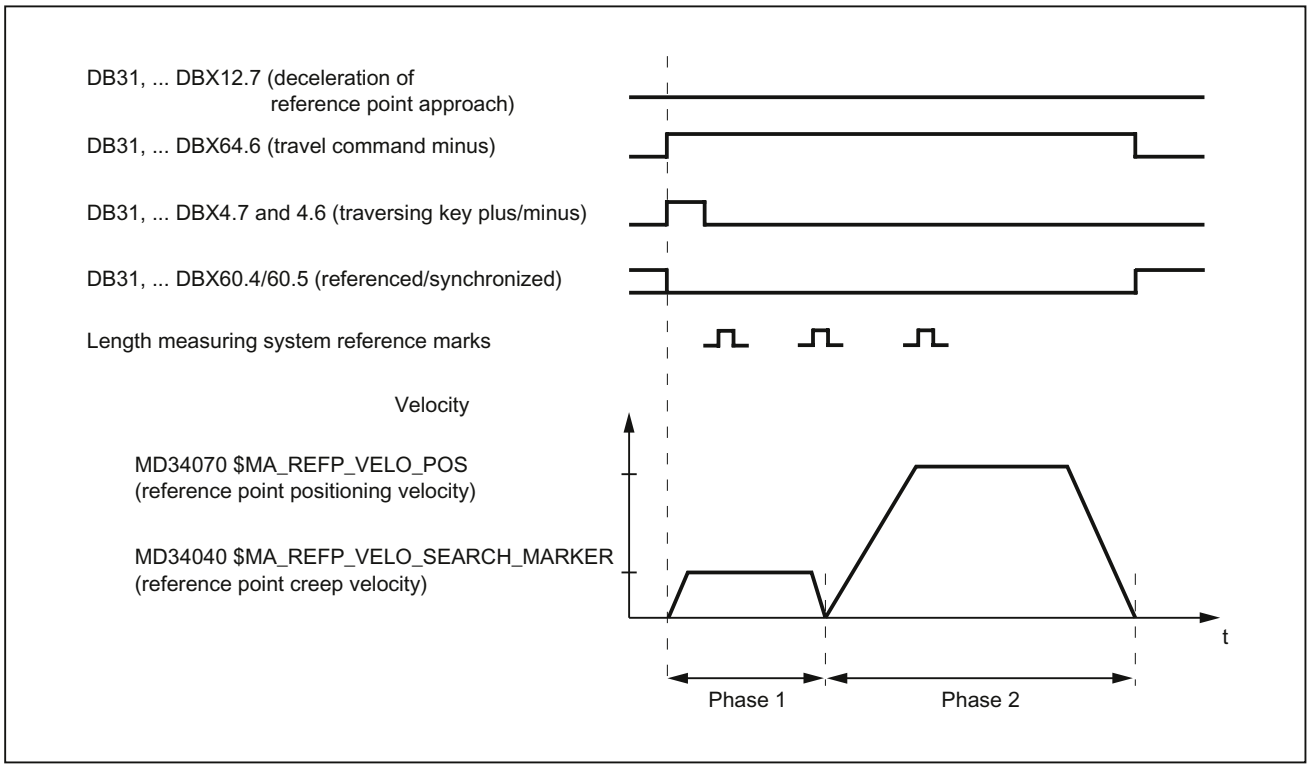


Figure 14-13 Distance-coded reference marks

14.6.4 Phase 1: Travel across the reference marks with synchronization

Phase 1: Start

To start the reference point approach, see Sections "Axisspecific referencing (Page 1321)" and "Channelspecific referencing (Page 1324)".

Reference cam

In measuring systems with distance-coded reference marks, reference cams are not required for the actual referencing action. For functional reasons, however, a reference cam is required for channel-specific reference point approach and reference point approach from the part program (G74) before the traversing range end of the machine axis.

Phase 1: Sequence

Sequence without contact witha reference cam

Once the reference point approaching process is started, the machine axis accelerates to the reference point shutdown speed set by means of parameter assignment:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Sequence when starting from the reference cam

If the machine axis is at the reference cam at the start of the reference point traversing, it accelerates to the parameterized reference point creep velocity against the parameterized reference point approach direction:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

MD34010 $MA_CAM_DIR_IS_MINUS (reference point approach in minus direction)

That ensures that the machine axis does not reach the travel range limit before it has crossed the parameterized number of reference marks.

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Sequence when contact is made with reference cam during referencing

Once the reference point approaching process is started, the machine axis accelerates to the reference point shutdown speed set by means of parameter assignment:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

Before the machine axis travels over the parameterized number of reference marks, it makes contact with the reference cam. It is then reversed and reference mark search is restarted in the opposite direction.

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Plausiblity check of the reference mark distance

An error occurs if, during reference point traversing for two subsequent reference marks, the NC determines a distance greater than twice the parameterized reference mark distance.

MD34300 $MA_ENC_REFP_MARKER_DIST (reference mark distance)

The machine axis will then traverse in opposite direction at half the parameterized reference point creep velocity (MD34040) and the search for reference mark is restarted.

If a faulty reference mark distance is detected again, the machine axis is stopped and the reference point traversing is aborted (alarm 20003 "fault in the measuring system").

Abort criterion

If the parameterized number of reference marks is not detected within the parameterized distance, the machine axis is stopped and reference point traversing is aborted.

MD34060 $MA_REFP_MAX_ MARKER_DIST (max. distance to the reference mark)

Features of Phase 1

After Phase 1 is successfully completed, the actual value system of the machine axis is synchronized.

14.6.5 Phase 2: Traversing to the target point

Phase 2: Start

Phase 2 is automatically started when Phase 1 has been completed without an alarm.

Initial situation:

- The machine axis is positioned directly behind the last of the parameterized number of reference marks.
- The actual value system of the machine axis is synchronized.

Phase 2: Sequence

In Phase 2, the machine axis completes reference point approach by traversing to a defined target position (reference point). This action can be suppressed in order to shorten the reference point approach:

MD34330 $MA_STOP_AT_ABS_MARKER = <value>

Value	Meaning
0	Travel to target position
1	No travel to target position

Travel to target position (normal case)

The machine axis accelerates to the parameterized reference point position velocity and travels to the parameterized target point (reference point):

MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)

MD34100 $MA_REFP_SET_POS (reference point value)

The machine axis is referenced. To identify this, the NC sets an interface signal for the measuring system that is currently active:

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

No travel to target position

The machine axis is now referenced. To identify this, the NC sets an interface signal for the measuring system that is currently active:

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

Features of Phase 2

Phase 2 will display different characteristics, depending on whether a reference cam is parameterized for the machine axis.

Machine axis without reference cam

MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference cam) = 0

Properties:

- Feed override active.
- The feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.

Machine axis with reference cam

MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference cam) = 1

Properties:

- Feedrate override is **not** active.

  Machine axis moves internally when feedrate override = 100%.

  If a feedrate override of 0% is specified, an abort occurs.
- The feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are **not** active.
- If the parameterized number of reference marks is not detected within the parameterized distance after the exit of the reference cam, the machine axis will be stopped.

  MD34060 $MA_REFP_MAX_ MARKER_DIST (max. distance to the reference mark)

Special features of rotary measuring systems

On rotary distance-coded measuring systems, the absolute position can only be determined uniquely within one revolution. Depending on the mechanical mounting of the encoder, the overtravel of the absolute position in the hardware does not always coincide with the traversing range of the rotary axis.

Special features of modulo rotary axes

With module rotary axes, the reference point position is mapped on the parameterized modulo range:

MD30330 $MA_MODULO_RANGE (size of the modulo range)

MD30340 $MA_MODULO_RANGE_START (start position of the modulo range)

Note

The reference point position is mapped onto the assigned (ghost) modulo range even with axis function "Determination of reference point position rotary, distance-coded encoder within the configured modulo range":

MD30455 $MA_MISC_FUNCTION_MASK (axis functions), BIT1 = 1

14.7 Referencing by means of actual value adjustment

14.7.1 Actual value adjustment to the referencing measurement system

Function

When actual value adjustment to the referencing measuring system is performed, the resulting absolute actual position after successful referencing of the measuring system of a machine axis is transferred directly to all other measuring systems of the machine axis, and the machine axis is designated as referenced:

DB31, ... DBB60.4 / 60.5 (referenced/synchronized 1/2) = 1

Advantage

When the machine axis switches from an explicitly referenced measuring system to the measuring system referenced by actual value adjustment, continuous servo control is assured (servo enable active) because the matched actual position prevents a sudden change in actual value.

Note

In order to improve positioning precision by determining the measuring-system-specific encoder fine information, we recommend explicitly re-referencing the measuring system previously referenced by actual value adjustment after switching over.

Activation

The activation of the actual value adjustment to the referencing measuring system is machine-specifically carried out via:

MD34102 $MA_REFP_SYNC_ENCS = 1

14.7.2 Actual value adjustment for measuring systems with distance-coded reference marks

Function

In order to improve positioning precision by determining the measuring-system-specific encoder fine information, we recommend explicitly re-referencing the measuring system previously referenced by actual value adjustment after switching over the measuring system.

If an encoder with distance-coded reference marks is used for the passive measuring system, referencing can be avoided if the following conditions are met:

1. Active measuring system: indirect measuring system (motor measuring system) with absolute encoder, for example
2. Passive measuring system: Direct measuring system with distancecoded reference marks
3. Travel movement of the machine axis with the referenced indirect measuring system before measuring system switchover in which the number of reference marks required for referencing are crossed. This automatically references the passive direct measuring system.

Parameterization

In addition to the specific machine data required to reference the individual measuring systems, the following machine data must be set:

- Enable actual value adjustment:

  MD34102 $MA_REFP_SYNC_ENCS = 1

- Direct measuring system with distancecoded reference marks:
  - MD34200 $MA_ENC_REFP_MODE[*measuring system*] = 3

    Distancecoded reference marks
  - MD30242 $MA_ENC_IS_INDEPENDENT[*measuring system*] = 2

    During actual value adjustment, the passive direct measuring system is adjusted to the actual position of the active indirect measuring system, but is not marked as referenced. After the parameterized number of reference marks have been crossed, the passive direct measuring system is automatically referenced. Referencing is performed in every operating mode.

Sequence

1. Initial situation: Neither of the measuring systems are referenced:
   DB31, ... DBX60.4 = 0 (referenced / synchronized 1)
   DB31, ... DBX60.5 = 0 (referenced / synchronized 2)
2. Reference the indirect measuring system according to the measuring system type:
   DB31, ... DBX60.4 = 1 (referenced / synchronized 1)
   DB31, ... DBX60.5 = 0 (referenced / synchronized 2)
3. Traverse the machine axis across the parameterized number of reference marks.
   This automatically references the direct measuring system:
   DB31, ... DBX60.4 = 1 (referenced / synchronized 1)
   DB31, ... DBX60.5 = 1 (referenced / synchronized 2)

14.8 Referencing in follow-up mode

Function

Incremental measuring systems and measuring systems with distance-coded reference marks can be referenced even when the machine axis is in follow-up mode. Prerequisite for this is the correct parameterization of the reference point approach according to the used measuring system (see Section "Referencing with incremental measurement systems (Page 1327)" and "Referencing with distance-coded reference marks (Page 1342)").

When referencing in follow-up mode the machine axis is moved not by the NC but by means of an external travel motion over the encoder zero mark and the parameterized number of distance-coded reference marks. The measuring system is referenced when the encoder zero mark or parameterized number of distance-coded reference marks are detected.

Note

Reproducibility of the referencing result

In NC-guided reference point approach, reproducibility of the referencing result is ensured through adherence to the assigned traverse velocities during the referencing operation. During referencing in follow-up mode, responsibility for achieving reproducibility of the referencing results lies with the machine manufacturer / user.

Unique zero mark

Referencing of an incremental measuring system is based on the explicit position of the encoder zero mark relative to the overall traversing range of the machine axis.

Because the reference cam signal is not evaluated by the NC during referencing in follow-up mode, unique identification of the reference point when referencing in follow-up mode will only result with:

- Only one encoder zero mark in the traversing range of the machine axis.
- Linear measuring systems with distance-coded reference marks.
- Modulo rotary axes (absolute position within one revolution).

Zero mark selection when several zero mark signals occur

If several encoder zero marks are detected in the traversing range of the machine axis due to machine-specific factors, e.g. reduction gear between encoder and load, a BERO must be mounted on the machine and connected via a digital input of the NCU interface in order to clearly determine the reference point.

Note

BERO signal: Zero mark selection

The processing of the BERO signal is performed exclusively in the drive. Connection and parameterization, see Section "Hardware signals (Page 1327)".

Enable

The "Referencing in follow-up mode" function is enabled with:

MD34104 $MA_REFP_PERMITTED_IN_FOLLOWUP = TRUE

Starting the referencing operation

If the machine axis is in follow-up mode (DB31, ... DBX61.3 == TRUE) at the start of reference point approach, the measuring system is referenced in follow-up mode.

If the machine axis is not operating in the follow-up mode at the start of reference point traversing, the "normal" from the NC-controlled reference point travels is carried out.

Referencing in follow-up mode can be started in the following modes:

- JOG-REF: Traversing keys
- AUTOMATIC: Part program command G74

Sequence of the referencing operation (JOG-REF mode)

1. Activate follow-up mode of machine axis:

   DB31, ... DBX1.4 (follow-up mode) = 0

   DB31, ... DBX2.1 (controller enable) = 0

2. Take into account activation of follow-up mode:

   DB31, ... DBX61.3 (follow-up active) = 1

3. Switch to JOG-REF mode.

4. External motion of machine axis across encoder zero mark or parameterized number of distance-coded reference marks. The referencing operation is started internally in the NC as soon as the machine axis is moved:

   DB31, ... DBX61.4 (axis/spindle stationary) = 0

5. The measuring system is referenced after the encoder zero mark or the assigned number of distance-coded reference marks have been successfully detected:

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

Aborting the reference operation

An active referencing operation can be aborted by:

- Deselecting follow-up mode
- NCK Reset

Response when measuring systems are already referenced

A measuring system that has already been referenced can only be re-referenced in AUTOMATIC mode using part program statement `G74`.

Sequence of referencing operation (AUTOMATIC mode)

1. Switch to AUTOMATIC mode.
2. Start the part program.
3. Activate follow-up mode of machine axis:

   DB31, ... DBX1.4 (follow-up mode) = 0

   DB31, ... DBX2.1 (controller enable) = 0
4. Take into account activation of follow-up mode:

   DB31, ... DBX61.3 (follow-up active) = 1
5. The referencing operation is started internally in the NC as soon as part program statement `G74` is processed.
6. External motion of machine axis across encoder zero mark or parameterized number of distance-coded reference marks.
7. The measuring system is referenced after the encoder zero mark or the assigned number of distance-coded reference marks have been successfully detected:

   DB31, ... DBX60.4/60.5 (referenced/synchronized 1/2) = 1
8. The block change occurs after the referencing operation has been successfully completed.

Aborting the reference operation

An active referencing operation can be aborted by:

- Deselecting follow-up mode
- NCK Reset

Response when measuring systems are already referenced

A measuring system that you have already referenced can be re-referenced.

14.9 Referencing with absolute encoders

14.9.1 Information about the adjustment

Machine axes with absolute encoder

The advantage of machine axes with absolute encoder is that after a one time adjustment procedure, the necessary reference point traversing with incremental measuring systems (e.g. build-up of control, de-selection of "Parking" of machine axes etc.) can be skipped and the actual value system of the machine axis can be immediately synchronized to the determined absolute position.

Adjustment

Adjustment of an absolute encoder involves matching the actual value of the encoder with the machine zero once and then setting it to valid.

The current adjustment status of an absolute encoder is displayed in the following axis specific machine data of the machine axis, to which it is connected:

MD34210 $MA_ENC_REFP_STATE (status of absolute encoder)

Value	Meaning
0	Encoder not calibrated
1	Encoder adjustment enabled
2	Encoder is calibrated

Adjustment methods

The following adjustment methods are supported:

- Adjustment by entering a reference point offset
- Adjustment by entering a reference point value
- Automatic adjustment with probe
- Adjustment with BERO

Readjustment

Readjustment of the absolute encoder is required after:

- Gear change between load and absolute encoder
- Removal/installation of the absolute encoder
- Removal/installation of the motor with the absolute encoder
- Data loss in the static NC memory
- Battery failure
- Setting actual value (PRESETON)

NOTICE

The controller can detect a required readjustment of the absolute encoder only during the following events:

- Gear change with change of gear ratio
- Addressing the zero-mark monitoring
- New encoder serial number after change of the absolute encoder

The controller then sets the status of the absolute encoder to "0":

MD34210 $MA_ENC_REFP_STATE = 0 (encoder not adjusted)

The following alarm is displayed:
Alarm 25022 "Axis <axis identifier> encoder <number> warning 0"
If the zero-mark monitoring responds, the following alarm is also displayed:
Alarm 25020 "Axis <axis identifier> zero-mark monitoring of active encoder"

In all other cases (e.g. PRESETON) it is the sole responsibility of the user to display the misalignment of the absolute encoder by **manually** setting the status to "0" and to carry out a readjustment.

⚠ WARNING

Data backup

During the back-up of machine data of a machine A, the encoder status of the machine axis (MD34210) is also backed up.

During loading of this data record into a machine B of the same type, e.g. in the context of a serial start-up or after a case of maintenance, the referenced machine axes will be automatically regarded as adjusted / referenced by the NC. It is the special responsibility of the machine manufacturer / user to undertake a readjustment in such cases.

See also explanations regarding machine data:

MD30250 $MA_ACT_POS_ABS (Absolute encoder position at the time of switch-off)

14.9.2 Calibration by entering a reference point offset

Function

During adjustment by entering the reference point offset, the difference between the position displayed on the operator interface and the true actual position in the machine is determined and made known to the NC as reference point offset.

Procedure

1. Determining the position of the machine axis with reference to the machine zero point via e.g.:
   - position measurement (e.g. laser interferometer)
   - Moving the machine axis to a known position (e.g., fixed stop)
2. Reading the displayed actual position of the machine axis on the operator interface.
3. Calculating the reference point offset (difference between the actual positions determined under point 1 and 2) and entering in machine data:

   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)
4. Marking the absolute value encoder as adjusted:

   MD34210 $MA_ENC_REFP_STATE = 2

   Note

   The encoder adjustment does not become active until the next time the encoder is activated (e.g., when the controller is powered up).
5. Initiate POWER ON reset.
6. Controlling the position of the machine axis displayed on the operator interface.

   Note

   Backlash compensation

   If backlash compensation is parameterized for a measuring system with absolute value encoder, the following must be observed:

   No backlash is permitted during machine axis travel to the adjusted machine position.

   Activate reference point offset permanently

   The entered reference point offset (MD34090) will be permanently active only after initial POWER ON - Reset. If the machine axis is moved after the absolute encoder adjustment without an interim POWER ON - Reset, the reference point offset entered in the machine data can be overwritten, for example, as part of internal overrun offset.

   Checking the actual position

   Following adjustment of the absolute encoder, we recommend that you verify the actual position of the machine axis the next time you power up the controller (POWER ON).

14.9.3 Adjustment by entering a reference point value

Function

During adjustment by entering the reference point value, the absolute position of the machine axis with reference to the machine zero point is determined by e.g.:

- Position measurement (e.g. laser interferometer)
- Moving the machine axis to a known position (e.g. fixed stop)

This determined position value will be made known to the NC as the reference point value. The NC then calculates the reference point offset from the difference between the encoder absolute value and the reference point value.

Procedure

1. Set reference mode to "Take over of the reference point value"

   MD34200 $MA_ENC_REFP_MODE = 0

2. Traversing machine axis in the JOG mode to the (e.g. Laser interferometer) position to be measured or already known (e.g. fixed stop).

   Note

   The machine axis can only be traversed in the direction enabled for referencing with the travel keys:

   MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point in minus direction)

   To avoid an invalid position because of backlash in the drive train, the known position must be approached at low velocity.

3. Communicate the position of the machine axis relative to machine zero to the NC as the reference point value:

   MD34100 $MA_REFP_SET_POS = *Position*

4. Releasing encoder adjustment:

   MD34210 $MA_ENC_REFP_STATE = 1

5. Activate NCK-Reset for acceptance of the entered machine data values.
6. Switch to JOG-REF mode.

7. Operate the travel key used for referencing in step 2.

   The machine axis does not move when the traversing key is actuated!

   The NC calculates the reference point offset from the entered reference point value and that given by the absolute value encoder. The result is entered into the machine data:

   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)

   The status of the absolute value encoder is set to "Encoder is adjusted":

   MD34210 $MA_ ENC_REFP_STATE = 2

   The actual value system of the machine axis is synchronized.

   The machine axis is now referenced. As identification, the NC sets the appropriate interface signal based on which measuring system is currently active:
   DB31, ... DBB60.4 / 60.5 (referenced/synchronized 1 / 2) = 1

8. Initiate POWER ON reset.

Note

Activate reference point offset permanently

The entered reference point offset (MD34090) will only be permanently active after POWER ON - Reset.

If the machine axis is moved after the absolute encoder adjustment without an interim POWER ON - Reset, the reference point offset entered in the machine data can be overwritten, for example, within internal overrun corrections.

Checking the actual position

Following adjustment of the absolute encoder, we recommend that you verify the actual position of the machine axis the next time you power up the controller (POWER ON).

14.9.4 Automatic calibration with probe

Function

In automatic adjustment with a probe, a known position in the machine is approached with the machine axis from a part program. The position value is stored in the NC as a reference point value. The position is reached when the probe switches, and the NC then calculates the reference point offset from the difference between the encoder value and reference point value.

Note

Part program for automatic adjustment

The part program for automatic adjustment using a probe must be created by the machine manufacturer / user for the specific requirements of the machine.

Freedom from collision

Because actual-value-related monitoring is not active for the machine axes being referenced, the machine operator must take special care to ensure that collisions do not occur in the machine while the machine axes are being moved!

Part program

The part program for automatic adjustment of absolute encoders with probe must perform the points listed below for each axis in the order indicated:

1. Approach the adjustment position of machine axis, which is detected from the probe response.

   The position must be approached several times from the same direction, but at a velocity which is gradually reduced on each approach, to ensure that the measured value obtained is as accurate as possible. The measured value is stored in system variable $AA_IM.

2. Calculating and writing the reference point offset:

   MD34090 $MA_REFP_MOVE_DIST_CORR = MD34100 $MA_REFP_SET_POS – $AA_IM

3. Set the absolute encoder status to "Encoder is adjusted":

   MD34210 $MA_ ENC_REFP_STATE = 2

Sequence

Proceed as follows for automatic adjustment with probe:

1. Enable part program start even for non-referenced machine axes:

   MD20700 $MC_REFP_NC_START_LOCK = 0

2. Enter the machine axis position relative to machine zero when probe is switched as the reference point value for all relevant machine axes:

   MD34100 $MA_REFP_SET_POS = reference point value

3. Activate NCK-Reset for the acceptance of the entered machine data values.
4. Start part program.
5. After completion of the part program, re-secure the partial program start for machine axes which are not referenced:

   MD20700 $MC_REFP_NC_START_LOCK = 1

6. Initiate POWER ON - Reset so that the reference point offset written by the part program is permanently active:

   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)

Note

Activate reference point offset permanently

The entered reference point offset (MD34090) will only be permanently active after POWER ON - Reset.

If the machine axis is moved after the absolute encoder adjustment without an interim POWER ON - Reset, the reference point offset entered in the machine data can be overwritten, for example, as part of internal overrun offset.

Checking the actual position

Following adjustment of the absolute encoder, we recommend that you verify the actual position of the machine axis the next time you power up the controller (POWER ON).

14.9.5 Adjustment with BERO

Function

For adjustment using BERO, a reference point approach to a defined machine position is performed the same as for incremental measuring systems. In this case the BERO replaces the encoder zero mark that the absolute encoder does not have. After successful completion of reference point approach, the NC automatically calculates the reference point offset from the difference between the encoder absolute value and the parameterized reference point value.

Parameterization

NC: Referencing mode

The referencing mode should be set to "Referencing of incremental, rotary or linear measuring systems: Zero pulse on the encoder track":

MD34200 $MA_ENC_REFP_MODE[<axis>] = 1

NC: Reference point value

The reference point value is parameterized via:

MD34100 $MA_REFP_SET_POS[<axis>] = <reference point value>

Drive: Equivalent zero mark

The digital input on the NCU interface to which the BERO is connected must be set in parameter p0494 or p0495.

Execution

Reference point approach can be started manually in JOG-REF mode or in AUTOMATIC or MDA mode via a part program (G74).

After successful completion of the reference point approach, the absolute encoder is adjusted and the actual-value system of the machine axis is synchronized.

As feedback for the PLC user program, the NC sets the NC/PLC interface signal for the machine axis, depending on the active measuring system:

DB31, ... DBB60.4/60.5 (referenced/synchronized 1/2) = 1

Note

If the BERO is removed after adjustment of the absolute encoder, the referencing mode must be re-parameterized to "Referencing with absolute encoder".

MD34200 $MA_ENC_REFP_MODE[<axis>] = 0

14.9.6 Reference point approach with absolute encoders

Traversing movement release

If for a mchine axis with adjusted absolute value encoder as active measuring system, reference point traversing is activated (manual in the mode JOG-REF or automatic according to part program instruction G74), the machine axis travels depending on the parameterized release traversing movement.

MD34330 $MA_REFP_STOP_AT_ABS_MARKER = <Value>

Value	Meaning
0	Traversing is enabled. When reference point approach is initiated, the machine axis moves to the reference point position. Reference point approach is completed when the reference point position is reached.
1	Traversing is **not** enabled. After the activation of the reference point travel, the machine axis does not travel and the reference point travel is immediately completed.

14.9.7 Reference point approach in rotary absolute encoders with external zero mark

Function

To be able to use the reference point approach with a zero mark, as is usual in incremental encoders (refer to Section "Referencing with incremental measurement systems (Page 1327)"), also with absolute encoders, the missing HW zero marks are created in software form once every encoder revolution, always at the same position within the rotation.

Difference compared to referencing with incremental encoders

An absolute encoder with replacement zero mark should not be considerd as a complete equivalent of an incremental encoder. All the properties of the absolute encoder are retained. The following table lists the different properties of incremental and absolute encoders:

Table 14- 1 Properties of incremental and absolute encoders

Property	Incremental encoder	Absolute encoder
Encoder type	MD30240 $MA_ENC_TYPE	
	= 1	= 4
Internal encoder position	MD30250 $MA_ACT_POS_ABS	
	Value is updated only in MD34210 ≥ 1	Value is updated only in MD30270 = 0
Traversing range extension	MD30270 $MA_ENC_ABS_BUFFERING	
	No effect	= 0 (default): Active
Reference point offset	MD34090 $MA_REFP_MOVE_DIST_CORR	
	Value input allowed	Value is updated exclusively via control
Supported referencing types	MD34200 $MA_ENC_REFP_MODE	
	= 1, 2, 3, 4, 5, 6, 7	= 0, 1, 2
Adjustment status	MD34210 $MA_ENC_REFP_STATE = 0, 1, 2	
	Automatic encoder misalignment during shut down while in motion.	Automatic encoder misalignment during parameter set change with position jump or during serial number change.
Absolute position modulo range	MD34220 $MA_ENC_ABS_TURNS_MODULO	
	= 0	= 1 - 4096
Encoder serial no.	MD34230 $MA_ENC_SERIAL_NUMBER	
	= 0	The value must be updated from the PLC during each encoder change, otherwise loss of adjustment plus alarm.
Transfer of series startup files	Without any restrictions.	Due to encoder properties MD30250, MD30270, MD34090, MD34210, MD34220, MD34230 only possible with limitations.
Activation time	0 seconds	several seconds
Zero mark	1 per encoder revolution	None
Zero-mark monitoring	Hardware	Software
Position after POWER ON without actual value buffering	0.0	Last position within MD34220.
	MD34210 = 0	MD30270 = 1
Position after POWER ON with actual value buffering	Last standstill position before deactivation.	Last position including small movements during POWER OFF.
	MD34210 = 1	MD30270 = 0
Referenced after POWER ON	depends on adjustment status	

Requirement

The function can be used only with rotary absolute encoders:

- MD31000 $MA_ENC_IS_LINEAR = 0
- MD30240 $MA_ENC_TYPE = 4

Parameterization

- reference point approach with zero marks:

  MD34200 $MA_ENC_REFP_MODE = 1
- A reference point offset should not be input in the following MD:

  MD34090 $MA_REFP_MOVE_DIST_CORR

  This MD describes, in connection with absolute encoders, the offset between machine and absolute encoder zero, and it therefore has a different meaning.
- The load-side zero mark search rate MD34040 $MA_REFP_VELO_SEARCH_MARKER should not exceed the limiting frequency of the absolute trace of the encoder MD36302 $MA_ENC_FREQ_LIMIT_LOW
  .

  If the speed is too high, absolute information cannot be read any more, and thus, no equivalent zero marks are generated.
- If no zero marks are found within:

  MD34060 $MA_REFP_MAX_MARKER_DIST

  otherwise, an alarm will be triggered.
- A start of the zero mark search with the override of a BERO (MD34200 = 5) is not supported. MD34200 = 0 is to be used as a equivalent.
- The following MD must be set if the absolute encoder retains even the referenced status through POWER OFF, besides the last position:

  MD34210 $MA_ENC_REFP_STATE = 2

Data backup and standard commissioning

Some properties of an absolute encoder restrict the transfer of series startup files to other machines. The following machine data must be checked and corrected if necessary after loading the series startup:

- MD30250 $MA_ACT_POS_ABS (internal encoder position)
- MD30270 $MA_ENC_ABS_BUFFERING (traversing range extension)
- MD34090 $MA_REFP_MOVE_DIST_CORR (absolute offset)
- MD34210 $MA_ENC_REFP_STATE (adjustment status)
- MD34220 $MA_ENC_ABS_TURNS_MODULO (Modulo range)
- MD34230 $MA_ENC_SERIAL_NUMBER (encoder serial number)

14.9.8 Automatic encoder replacement detection

Function

Automatic encoder replacement detection is required for absolute encoders in order to detect if the encoder has been replaced and therefore needs to be readjusted.

The NC reads the encoder-specific serial number of the encoder from the drive every time the control is powered up. If the serial number has changed the NC resets the encoder status to "Encoder not calibrated".

MD34210 $MA_ENC_REFP_STATE = 0

The status of the measuring system is indicated as "Not referenced":

DB31, ... DBB60.4 / 60.5 (referenced/synchronized 1/2) = 0

Serial number display

The NC stores the serial numbers read in the build-up specific to the machine in the machine data:

MD34230 $MA_ENC_SERIAL_NUMBER (encoder serial number)

Note

Currently, only the serial numbers of absolute encoders with an EnDat interface can be read. For all other encoders the display shows that no serial number has been read.

Automatic encoder replacement detection can therefore only be used with the specified encoder types.

Avoiding readjustments

In some special cases, for example, when a machine axis (built-on rotary axes) is removed and then mounted again, readjustment is not necessary / desirable.

To avoid readjustment, zero must be parameterized as a serial number to be ignored for the measuring system of the machine axis in question.

MD34232 $MA_EVERY_ENC_SERIAL_NUMBER = 0

If the NC now reads zero as the serial number, the encoder status is not reset and the serial number indicated in the machine data is kept.

Example sequence of operation:

1. The NC reads the serial number of the absolute encoder for the measuring system of the machine axis in question and the serial number is not equal to zero.
2. The absolute encoder is calibrated in the correct manner.
3. When the controller is powered up subsequently, the NC reads "zero" as the serial number of the absolute encoder.

   Serial number "zero" is ignored and the encoder status remains the same, that is "calibrated".
4. When the controller is powered up, the NC again reads the serial number it read under Item 1 and that is still indicated in the machine data. The encoder status continues to be "Adjusted".

Note

PROFIBUS drives

As not every drive connected via PROFIBUS-DP is able to deliver the encoder serial number in time for build-up of control or at all, the range of the encoder serial number with PROFIBUS drives is pre-set with zero to avoid unnecessary new NC internal adjustments:

MD34232 $MA_EVERY_ENC_SERIAL_NUMBER = 0

A manual parameterizing to 1 is ineffective.

14.9.9 Enabling the measurement system

The measuring system of a machine axis is activated in the following cases:

- Power up of the control (POWER ON)
- Activation of the measuring system via interface signal (deselection of "parking"):

  DB31, ... DBB1.5 / 1.6 (position measuring system 1/2)

  DB31, ... DBB2.1 (servo enable)
- Violation of the assigned encoder limit frequency (spindles):

  MD36300 $MA_ENC_FREQ_LIMIT

When the measuring system is activated, the NC synchronizes the actual value system of the machine axis with the current absolute value. Traversing is disabled during synchronization for axes but not for spindles.

Parameterizing the encoder limit frequency (spindles)

The EQN 1325 absolute encoder made by Heidenhain has an incremental track and an absolute track.

If a spindle is driven at a speed above the encoder limit frequency of the incremental track, the substantially lower limit frequency of absolute track must be parameterized as the encoder limit frequency.

MD36300 $MA_ENC_FREQ_LIMIT

Otherwise an incorrect absolute position would be read because the parameterized encoder limit frequency is not reached when the measuring system is activated. This would cause a position offset in the actual value system of the machine axis.

Determining the encoder limit frequency

The encoder limit frequency to be parameterized is derived from the smaller of the two following limit speeds:

- Encoder

  The limit speed or encoder limit frequency is listed in the data sheet of the encoder (e.g., limit speed = 2000 [rpm])

- NC

  Due to the NC-internal evaluation process, the maximum limit speed for which error-free calculation of the absolute value by the NC is possible is 4 encoder revolutions per interpolation cycle.

  For an interpolation cycle of, for example, 12 ms: Limit speed = 4 / 12 ms = 20,000 rpm

  The limiting frequency corresponding to the limiting speed is calculated to be:

$$MD36300 = \frac{4 * MD31020}{MD10050 * MD10070}$$

MD31020 $MA_ENC_RESOL (Encoder lines per revolution)

MD10050 $MN_SYSCLOCK_CYCLE_TIME (System clock cycle)

MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO (Factor for interpolator cycle)

Note

The position control switching speed relevant for spindles is set according to the encoder limiting frequency of the absolute value encoder of the spindle:

MD35300 $MA_SPIND_POSCTRL_VELO (position control switching speed)

MD36300 $MA_ENC_FREQ_LIMIT (Encoder limit frequency)

14.9.10 Referencing variants not supported

The following referencing variants are not supported when used with absolute encoders:

- Referencing/calibrating with encoder zero mark
- Distance-coded reference marks
- BERO with two-edge evaluation

14.10 Automatic restoration of the machine reference

Without a defined machine reference when traversing machine axes, no position-dependent functions such as transformations or tool frames can be executed. In various machine situations, these functions must be available immediately with the encoder activation, e.g. after the control is turned on or after terminating "parking (Page 69)", to traverse the axes. However, the machine axes should not or cannot be traversed again for referencing.

Absolute encoders

For measuring systems with adjusted absolute encoders, the machine reference is restored immediately without any additional measures when the encoder value is read.

Incremental encoders

With incremental measuring systems, the machine reference can be restored without traversing the axes through "Automatic referencing" or "Restoration of the actual position".

Supplementary conditions

⚠ WARNING

During the time in which the measuring system of the machine axis is switched off, it must **not** be moved mechanically. Otherwise this results in an offset between the last buffered actual position and the real actual position of the machine axis. This would lead to an incorrect synchronization of the measuring system resulting in danger to personnel and machine.

The machine manufacturer must provide such measures as holding brakes, etc. on the machine so that the actual position is not changed, and this must be ensured by the user. The responsibility for this rests exclusively with the machine manufacturer / user.

If axis motion cannot be prevented mechanically in the shutdown state, either an absolute encoder must be used or the axis must be referenced again with reference point approach after switching on.

NOTICE

SMExx sensor modules

Automatic referencing or restoration of the actual position to the last buffered position after restarting the control is only possible in conjunction with SM**E**xx (externally mounted) sensor modules. When using SM**C**xx (cabinet) or SM**I**xx (integrated) sensor modules, the actual position **cannot** be restored after restarting the control (power on). The measuring system of the machine axis must be referenced again.

14.10.1 Automatic referencing

Function

During automatic referencing, the actual position of the active measuring system of the machine axis is set to the last buffered position and "referenced" set as encoder state after switching on the control. This makes it possible to start programs in the AUTOMATIC and MDI modes directly after run-up of the control.

Requirements

- The active measuring systems when the control is switched on must already have been referenced once before switching off.
- At the time the control is switched off, the machine axis must be at standstill with "Exact stop fine" (DB31, ... DBX60.7 == 1) are located.

Note

If the machine axis is **not** at standstill with "Exact stop fine" when switching off, the actual position will be initialized with " 0" when switching on. "Not referenced" is displayed as the encoder state.

Parameter assignment

The automatic referencing is enabled by setting the encoder state to "Automatic referencing is enabled, but the encoder has not been referenced":

MD34210 $MA_ENC_REFP_STATE[<encoder>] = **1**

After the measuring system has been referenced, the encoder state displays that automatic referencing will be executed the next time the encoder is activated:

MD34210 $MA_ENC_REFP_STATE[<encoder>] == **2**

NC/PLC interface signals

After automatic referencing, the encoder state "Referenced" is displayed for the active measuring system:

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

Supplementary conditions

Encoder activation with MD34210 $MA_ENC_REFP_STATE[<encoder>] == 1

An encoder state equal to "1" at the time of the encoder activation means that "Automatic referencing" has been enabled. However, the measuring system has either not been referenced yet or the machine axis was not switched off at standstill in the "Exact stop fine" state. The following is set for the machine axis or the active measuring system:

- Actual position = 0
- Active measuring system, encoder state = "Not referenced":

  DB31, ... DBX60.4 / .5 = **0** (referenced/synchronized 1/2)

References

Description of Functions, Basic functions, Section "R1 Referencing" > "Referencing with incremental measurement systems (Page 1327)" > ""

14.10.2 Restoration of the actual position

Function

When restoring the actual position to the last buffered position, the encoder state of the active measuring system is set to "**Restored**". The axis can only be traversed manually.

AUTOMATIC mode

To enable NC start for the automatic execution of programs in the AUTOMATIC mode, the measuring system of the machine axis must be re-referenced.

MDI mode and overstore

In the MDI mode and for the overstore function, machining can also be performed, without referencing the axes, with restored positions. To do this, NC start with restored positions must be enabled explicitly for a specific channel:

MD20700 $MC_REFP_NC_START_LOCK = 2

Requirement

The active measuring systems when the control is switched on must already have been referenced once before switching off.

Parameter assignment

Release: Restoration of the actual position

The enable to restore the actual position is performed by setting the encoder state to "The last buffered axis position before switching off will be restored, no automatic referencing":

MD34210 $MA_ENC_REFP_STATE[<encoder>] = **3**

Release: NC START for "MDI" and "Overstore" modes

The enable of NC START for execution of part programs or part program blocks in the "MDI" and "Overstore" modes with the state "Position restored" is performed via:

MD34110 $MA_REFP_CYCLE_NR **≠ -1** (axis sequence for channel-specific referencing)

MD20700 $MC_REFP_NC_START_LOCK = **2** (NC START lock without reference point)

NC/PLC interface signals

The restored actual position is not considered to be equivalent to an actual position after reference point approach. Therefore, the state "Position restored" and not "Referenced/synchronized" is displayed for the measuring system of the machine axis.

Actual position restored:

- DB31, ... DBX60.4/.5 = **0** (referenced/synchronized 1/2)
- DB31, ... DBX71.4/.5 = **1** (**position restored**, encoder 1/2)

Measuring system referenced:

- DB31, ... DBX60.4/.5 = 0 → **1** (**referenced/synchronized** 1/2)
- DB31, ... DBX71.4/.5 = 1 → **0** (position restored, encoder 1/2)

Note

The monitoring of the traversing range limits (software limit switches, working area limitation, etc.) is already active in the "Position restored" state.

Supplementary conditions

Spindles

If the encoder limit frequency is exceeded, a spindle is reset to the "Not referenced/synchronized" state:

- DB31, ... DBX60.4/.5 = 1 → **0** (referenced/synchronized 1/2)
- DB31, ... DBX71.4/.5 = 1 → **0** (position restored, encoder 1/2)

References

Description of Functions, Basic functions, Section "R1 Referencing" > "Referencing with incremental measurement systems (Page 1327)" > ""

14.11 Supplementary conditions

14.11.1 Large traverse range

Linear axes with a traversing range > 4096 encoder revolutions, rotatory absolute encoder EQN 1325 and a parameterized absolute encoder range of MD34220 $MA_ENC_ABS_TURNS_MODULO = 4096:

The maximum possible travel range corresponds to that of incremental encoders.

Endlessly turning rotary axes with absolute encoders:

- Any number of integer transmission ratios are permitted.
- We recommend that you parameterize endlessly turning rotary axes with absolute encoders as modulo rotary axes (traversing range 0...360 degrees):
  MD30310 $MA_ROT_IS_MODULO = 1

  Otherwise, the rotary axis may require a very large traversing path to reach absolute zero when the measuring system is activated.

Machine axes with absolute encoders:

In order that the controller correctly determines the current actual position after the restart of the measuring system, the machine axis may only be moved less than half the absolute encoder range when the measuring system is switched off:

MD34220 $MA_ENC_ABS_TURNS_MODULO

Notes on uniqueness of encoder positions

Note

Linear absolute encoders

The absolute value of linear position encoders, e.g. Heidenhain LC181, is always unique for the scale lengths available.

Rotary absolute encoders

The absolute value of rotary absolute encoders is only unique within the range of the specific maximum encoder revolutions.

For example, the EQN 1325 rotary absolute encoder by Heidenhain supplies a unique absolute value in the range of 0 to 4,096 encoder revolutions.

Depending on how the encoder is connected that will result in:

- Rotary axis with encoder on load: 4096 load revolutions
- Rotary axis with encoder on motor: 4096 motor revolutions
- Linear axis with encoder on motor: 4096 motor revolutions

Example:
An EQN 1325 rotary absolute encoder is mounted on the motor of a linear axis. For an effective leadscrew pitch of 10 mm this will result in a unique absolute value within the travel range -20.48 to +20.48 m.

14.12 Data lists

14.12.1 Machine data

14.12.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
11300	JOG_INC_MODE_LEVELTRIGGRD	INC/REF in jog/continuous mode

14.12.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20700	REFP_NC_START_LOCK	NC start disable without reference point

14.12.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30200	NUM_ENCS	Number of encoders
30240	ENC_TYP	Actual value encoder type
30242	ENC_IS_INDEPENDENT	Encoder is independent
30250	ACT_POS_ABS	Absolute encoder position at time of deactivation.
30270	ENC_ABS_BUFFERING	Absolute encoder: Traversing range extension
30300	IS_ROT_AX	Rotary axis/spindle
30310	ROT_IS_MODULO	Modulo conversion for rotary axis / spindle
30330	MODULO_RANGE	Magnitude of the modulo range
30340	MODULO_RANGE_START	Starting position of modulo range
30355	MISC_FUNCTION_MASK	Axis functions
31122	BERO_DELAY_TIME_PLUS	BERO delay time in plus direction
31123	BERO_DELAY_TIME_MINUS	BERO delay time in minus direction
34000	REFP_CAM_IS_ACTIVE	Axis with reference cam
34010	REFP_CAM_DIR_IS_MINUS	Reference point approach in minus direction
34020	REFP_VELO_SEARCH_CAM	Homing approach velocity
34030	REFP_MAX_CAM_DIST	Maximum distance to reference cam
34040	REFP_VELO_SEARCH_MARKER	Reference point creep velocity
34050	REFP_SEARCH_MARKER_REVERSE	Direction reversal to reference cam
34060	REFP_MAX_MARKER_DIST	Maximum distance to reference mark; Max. distance to 2 reference mark for distance-coded scales

Number	Identifier: $MA_	Description
34070	REFP_VELO_POS	Reference point positioning velocity
34080	REFP_MOVE_DIST	Reference point distance/destination point for distancecoded system
34090	REFP_MOVE_DIST_CORR	Reference point/absolute offset, distancecoded
34092	REFP_CAM_SHIFT	Electronic reference cam shift for incremental measurement systems with equidistant zero marks.
34093	REFP_CAM_MARKER_DIST	Reference cam/reference mark distance
34100	REFP_SET_POS	Reference point value
34102	REFP_SYNC_ENCS	Actual value adjustment to the referencing measurement system
34104	REFP_PERMITTED_IN_FOLLOWUP	Enable referencing in followup mode
34110	REFP_CYCLE_NR	Axis sequence for channelspecific Homing
34120	REFP_BERO_LOW_ACTIVE	Polarity change of the BERO cam
34200	ENC_REFP_MODE	Referencing mode
34210	ENC_REFP_STATE	Status of absolute encoder
34220	ENC_ABS_TURNS_MODULO	Absolute encoder range for rotary encoders
34230	ENC_SERIAL_NUMBER	Encoder serial number
34232	EVERY_ENC_SERIAL_NUMBER	Expansion of encoder serial number
34300	ENC_REFP_MARKER_DIST	Basic distance of reference marks for distance-coded encoders
34310	ENC_MARKER_INC	Interval between two reference marks with distance-coded scales
34320	ENC_INVERS	Linear measuring system inverse to machine system:
34330	REFP_STOP_AT_ABS_MARKER	Distancecoded linear measuring system without destination point
35150	SPIND_DES_VELO_TOL	Spindle speed tolerance
36302	ENC_FREQ_LIMIT_LOW	Encoder limit frequency resynchronization
36310	ENC_ZERO_MONITORING	Zero mark monitoring

14.12.2 Signals

14.12.2.1 Signals to BAG

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode group RESET	DB11.DBX0.7	DB3000.DBX0.7
Machine function REF	DB11.DBX1.2	DB3000.DBX1.2

14.12.2.2 Signals from BAG

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

14.12.2.3 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate referencing	DB21, ...DBX1.0	DB3200.DBX1.0
OEM channel signal (HMI → PLC) REF	DB21, ...DBX28.7	-

14.12.2.4 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referencing active	DB21, ...DBX33.0	DB3300.DBX1.0
Reset	DB21, ...DBX35.7	DB3300.DBX3.7
All axes referenced	DB21, ...DBX36.2	DB3300.DBX4.2

14.12.2.5 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Follow-up mode (request)	DB31, ...DBX1.4	DB380x.DBX1.4
Position measuring system 1 / position measuring system 2	DB31, ...DBX1.5-6	DB380x.DBX1.5-6
Reference point value 1 to 4	DB31, ...DBX2.4-7	DB380x.DBX2.4-7
Traversing keys minus/plus	DB31, ...DBX4.6-7	DB380x.DBX4.6-7
Deceleration of reference point approach	DB31, ...DBX12.7	DB380x.DBX1000.7

14.12.2.6 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced, synchronized 1/2	DB31, ...DBX60.4-5	DB390x.DBX0.4-5
Follow up active	DB31, ...DBX61.3	DB390x.DBX1.3
Traversing command minus/plus	DB31, ...DBX64.6-7	DB390x.DBX4.6-7
Restored 1/2	DB31, ...DBX71.4-5	DB390x.DBX11.4-5

S1: Spindles

15

15.1 Brief Description

The primary function of a spindle is to set a tool or workpiece in rotary motion in order to facilitate machining.

Depending on the type of machine, the spindle must support the following functions in order to achieve this:

- Input of a direction of rotation for the spindle (M3, M4)
- Input of a spindle speed (S, SVC)
- Spindle stop, without orientation (M5)
- Spindle stop with orientation / Spindle positioning
  (SPOS, M19 and SPOSA)
- Gear change (M40 to M45)
- Spindleaxis functionality (spindle becomes rotary axis and vice versa)
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck (G331, G332)
- Tapping with compensating chuck (G63)
- Revolutional feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- Programmable spindle speed limits (G25, G26, LIMS=)
- Position encoder assembly on the spindle or on the spindle motor
- Spindle monitoring for min. and max. speed as well as
  max. encoder limit frequency and end point monitoring of spindle
- Switching the position control (SPCON, SPCOF, M70) on/off
- Programming of spindle functions:
  - From the part program
  - Via synchronized actions
  - Via PLC with FC18 or via special spindle interfaces for simple spindle activation

15.2 Modes

15.2.1 Overview

Spindle modes

The spindle can have the following modes:

- Control mode
- Oscillation mode
- Positioning mode
- Synchronous mode, synchronous spindle

  References:
  Function Manual, Extension Functions, Synchronous Spindle (S3)
- Rigid tapping

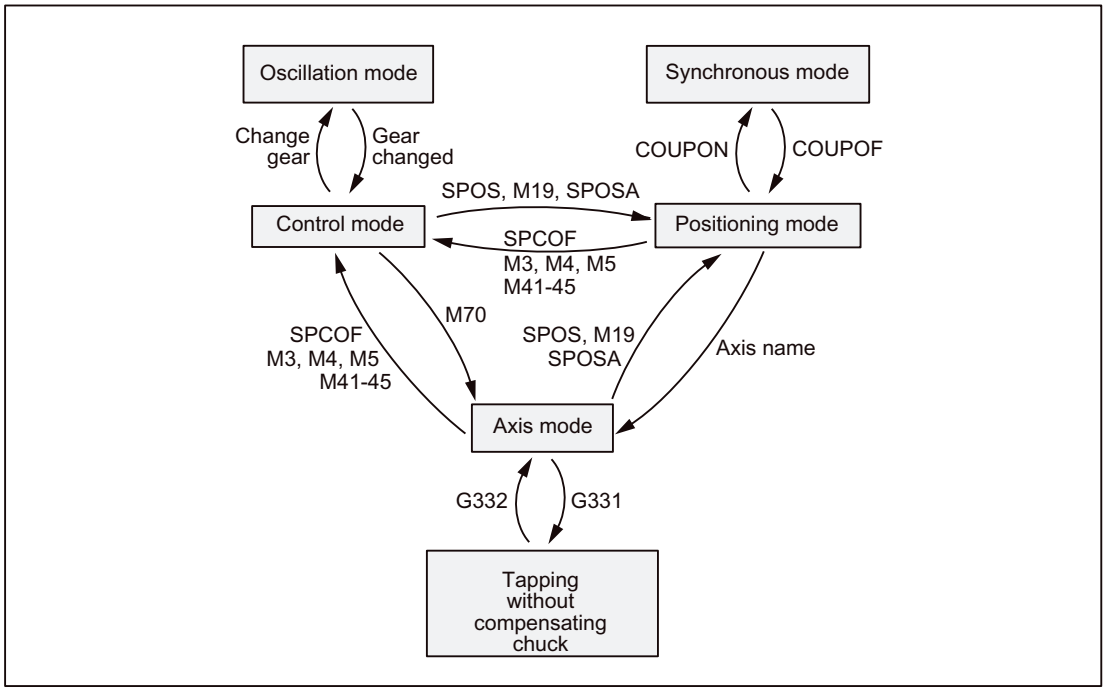
  References:
  Programming Manual, Fundamentals; Chapter: Motion commands

Axis mode

The spindle can be switched from spindle mode to axis mode (rotary axis) if the same motor is used for spindle and axis operation.

15.2.2 Mode change

Switching between spindle and axis operation can be done as follows:



- Control mode → Oscillation mode

  The spindle changes to oscillation mode if a new gear stage has been specified using automatic gear step selection (M40) in conjunction with a new S value or by M41 to M45. The spindle only changes to oscillation mode if the new gear step is not equal to the current actual gear step.

- Oscillation mode → Control mode

  When the new gear is engaged, the interface signal:
  DB31, ... DBX84.6 (Oscillation mode)
  is reset and the interface signal:
  DB31, ... DBX16.3 (Gear changed)
  is used to go to control mode.
  The last programmed spindle speed (S value) is reactivated.

- Control mode → Positioning mode

  To stop the spindle from rotation (M3 or M4) with orientation or to reorient it from standstill (M5), SPOS, M19 or SPOSA are used to switch to positioning mode.

- Positioning mode → Control mode

  M3, M4 or M5 are used to change to control mode if the orientation of the spindle is to be terminated. The last programmed spindle speed (S value) is reactivated.

- Positioning mode → Oscillation mode

  If the orientation of the spindle is to be terminated, M41 to M45 can be used to change to oscillation mode. When the gear change is complete, the last programmed spindle speed (S value) and M5 (control mode) are reactivated.

- Positioning mode → Axis mode

  If a spindle was stopped with orientation, the assigned axis name is used to program a change to axis mode. The gear step is retained.

- Control mode → Axis mode

  Switching from control mode to axis mode can be also achieved by the programming of M70. In this case, a rotating spindle is decelerated in the same way as for M5, position control activated and the zero parameter set selected.

- Axis mode → Control mode

  To terminate axis mode, M3, M4 or M5 can be used to change to control mode. The last programmed spindle speed (S value) is reactivated.

- Axis mode → Oscillation mode

  To terminate axis mode, M41 to M45 can be used to change to oscillation mode (only if the programmed gear step is not the same as the actual gear step). When the gear change is complete, the last programmed spindle speed (S value) and M5 (control mode) are reactivated.

15.2.3 Control mode

When open-loop control mode?

The spindle is in open-loop control mode with the following functions:

- Constant spindle speed:
  - S... M3/M4/M5 and G93, G94, G95, G97, G971
  - S... M3/M4/M5 and G33, G34, G35
  - S... M3/M4/M5 and G63
- Constant cutting speed:
  - G96/G961 S... M3/M4/M5

The spindle need not be synchronized.

Requirements

A spindle position actual value encoder is absolutely essential for M3/M4/M5 in connection with:

- Revolutional feedrate (G95)
- Constant cutting speed (G96, G961, G97, G971)
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck (G331, G332)
- Activate position control (SPCON, M70)

A spindle position actual value encoder is **not** required for M3/M4/M5 in connection with:

- Inverse-time feedrate coding (G93)
- Feedrate in mm/min or inch/min (G94)
- Tapping with compensating chuck (G63)

Speed control mode

Speed control mode is particularly suitable if a constant spindle speed is required, but the position of the spindle is not important (e.g. constant milling speed for even appearance of the workpiece surface).

- Speed control mode is activated in the part program with `M3`, `M4`, `M5` or with `SPCOF`.
- The following NC/PLC interface signal is set:

  DB31, ... DBX84.7 (control mode)
- NC/PLC IS:
  DB31, ... DBX61.5 (position controller active)
  is reset if position control is not used.
- Acceleration in speed control mode is defined independently of the gear stage in the machine data:

  MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL

  The value should reflect the physical circumstances, if possible.

Position control mode

Position control is particularly suitable if the position of the spindle needs to be tracked over a longer period or if synchronous spindle setpoint value linkage is to be activated.

- Position control mode is switched on in the part program with: `SPCON(<spindle number>)`
- The following NC/PLC interface signal is set:

  DB31, ... DBX61.5 (position controller active)
- Acceleration in position control mode is defined independent of the gear stage in the machine data:

  MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Independent spindle reset

The spindle response after a reset or the end of the program (M2, M30) is set with the machine data:

MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (individual spindle reset)

Value	Meaning
0	When the spindle is reset or at the end of the program, the spindle immediately decelerates to a stop at the active acceleration rate. The last programmed spindle speed and direction of rotation are deleted.
1	Upon reset or at the end of the program, the last programmed spindle speed (S-value) and the last programmed direction of spindle rotation (M3, M4, M5) are retained. The spindle is not braked.

If prior to reset or end of program the constant cutting speed (G96, G961) is active, the current spindle speed (in relation to 100% spindle override) is internally accepted as the spindle speed last programmed.

The spindle can only be stopped with the NC/PLC interface signal:

DB31, ... DBX2.2 (delete distance-to-go / spindle reset)

The direction of rotation is deleted in the event of all alarms generating a spindle quick stop. The last programmed spindle speed (S value) is retained. Once the source of the alarm has been eliminated, the spindle must be restarted.

Spindle actual speed display and spindle behavior with G96, G961

DB31, ... DBX61.4 (axis/spindle stationary)

The speed at which the spindle is deemed to be "stationary" is set with the machine data:

MD36060 $MA_STANDSTILL_VELO_TOL

The value should be measured in such a way that the following NC/PLC interface signal is reliably present at a standstill:

DB31,... DBX61.4 (axis/spindle stationary)

If DB31,... DBX61.4 (axis/spindle stationary) is signaled and there is no closed-loop position control active for the spindle, an actual speed of zero is displayed at the user interface, and zero is read with the system variable $AA_S[<n>].

Spindle response at constant cutting speed G96, G961

- At the start of machining (transition from G0 to Gx) and after NC stop, G60 (exact stop, modal) and G9 (exact stop, non-modal) the system waits until the actual speed has reached the speed setpoint tolerance range before starting the path.

  DB31, ... DBX83.5 (nact = nset)

- The NC/PLC IS:
  DB31, ... DBX83.5 (nact = nset)
  and
  DB31, ... DBX83.1 (setpoint speed limited) are also set to defined values
  even if significant speed changes are specified (transverse axis travels towards position 0).

- If the speed drops below the minimum speed
  or when NC/PLC IS:
  DB31, ... DBX61.4 (axis/spindle stationary)
  is detected, NC/PLC IS:
  DB31, ... DBX83.5 (nact = nset)
  is reset (e.g. for an emergency machine strategy).

- A path operation which has started (G64, rounding), is not interrupted.

In addition, the spindle response is influenced by the following machine data:

MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START (feed enable with spindle in setpoint range).

Spindle behavior at the end of gear stage change

- NC/PLC IS:
  DB31, ... DBX16.3 (gear changed)
  informs the NC that the new gear stage
  (NC/PLC IS DB31, ... DBX16.0-16.2 (actual gear stage A to C))
  applies and oscillation mode is terminated.

  In this case, it does not matter whether NC/PLC IS:
  DB31, ... DBX18.5 (oscillation mode)
  is still set.

  The actual gear stage should correspond to the set gear stage.

  The actual gear stage signaled is relevant for selection of the parameter set.

- Once the gear stage change (GSW) has been acknowledged via the PLC (DB31, ... DBX16.3), the spindle is in speed control mode (DB31, ... DBX84.7 = 1).

  If a direction of rotation (M3, M4, M5 or FC18: "Start spindle rotation") or a spindle speed (S value) was programmed before the gear stage change, then the last speed and direction of rotation will be reactivated after the gear stage change.

15.2.4 Oscillation mode

Oscillation mode is activated for the spindle during the gear step change.

The mode of operation is described in detail in the topic "Gear step change with oscillation mode (Page 1443)".

15.2.5 Positioning mode

15.2.5.1 General functionality

When is positioning mode used?

The spindle positioning mode stops the spindle at the defined position and activates the position control, which remains active until it is de-activated.

For the following functions the spindle is in positioning mode:

- SPOS[<n>]=...
- SPOS[<n>]=ACP(...)
- SPOS[<n>]=ACN(...)
- SPOS[<n>]=AC(...)
- SPOS[<n>]=IC(...)
- SPOS[<n>]=DC(...)
- SPOSA[<n>]=ACP(...)
- SPOSA[<n>]=ACN(...)
- SPOSA[<n>]=AC(...)
- SPOSA[<n>]=IC(...)
- SPOSA[<n>]=DC(...) equal to SPOSA[<n>]=...
- M19 or M[<n>]=19

The address extension [<n>] with <n> = spindle number may not apply for the main spindle.

SPOS[<n>]=AC(...)

Spindle positioning to an absolute position (0 to 359.999 degrees). The positioning direction is determined either by the current direction of spindle rotation (spindle rotating) or the distance-to-go.

SPOS[<n>]=IC(...)

Spindle positioning to an incremental position (+/- 999999.99 degrees) in relation to the last programmed position. The positioning direction is defined by the sign of the path to be traversed.

SPOS[<n>]=DC(...)

Spindle positioning across the shortest path to an absolute position (0 to 359.999 degrees). The positioning direction is determined either by the current direction of spindle rotation (spindle rotating) or automatically by the control (spindle stationary).

SPOS[<n>]=...

Same functional sequence as SPOS [<n>]=DC(...).

SPOS[<n>]=ACP(...)

Approaches the position from the positive direction.

When positioning from a negative direction of rotation, the speed is decelerated to zero and accelerated in the opposite direction to execute the positive approach.

SPOS[<n>]=ACN(...)

Approaches the position from the negative direction.

When positioning from a positive direction of rotation, the speed is decelerated to zero and accelerated in the opposite direction to execute the negative approach.

M19 (DIN 66025)

M19 can be used to position the spindle. The position and the position approach mode are read here from the following setting data:

SD43240 $SA_M19_SPOS[<n>] (spindle position for spindle positioning with M19)

SD43250 $SA_M19_SPOSMODE[<n>] (spindle position for spindle positioning with M19)

The positioning options of M19 are identical to those of:

SPOS = <approach mode> <position/path>

M19 is output as an auxiliary function to the NC/PLC interface as an alternative to M3, M4, M5, and M70. The M19 block remains active in the interpolator for the duration of positioning (like SPOS).

Part programs using M19 as a macro (e.g. `DEFINE M19 AS SPOS = 0`) or as a subprogram, continue to remain executable. For the sake of compatibility with previous controls, the internal processing of M19 (NCK positions the spindle) can be disabled as shown in the following example:

```
MD22000 $MC_AUXFU_ASSIGN_GROUP[0] = 4          ; Auxiliary function group: 4
MD22010 $MC_AUXFU_ASSIGN_TYPE[0] = "M"         ; Auxiliary function type: "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[0] = 0      ; Auxiliary functions Extension: 0
MD22030 $MC_AUXFU_ASSIGN_VALUE[0] = 19         ; Auxiliary function value: 19
```

Implicitly generated auxiliary function M19

To achieve uniformity in terms of how M19 and SPOS or SPOSA behave at the NC/PLC interface, auxiliary function M19 can be output to the NC/PLC interface in the event of SPOS and SPOSA.

The two following options are available for activating this function:

- Channel-specific activation for all the spindles in the channel via the machine data:

  MD20850 $MC_SPOS_TO_VDI (Output of M19 to the PLC with SPOS/SPOSA)

Bit	Value	Meaning
0	0	If bit 19 is also set to "0" in the MD35035 $MA_SPIND_FUNCTION_MASK, no auxiliary function M19 is generated in SPOS and SPOSA. This therefore eliminates the acknowledgement time for the auxiliary function.
	1	The auxiliary function M19 is generated and output to the PLC during the programming of SPOS and SPOSA in the part program. The address extension corresponds to the spindle number.

- Spindle-specific and cross-channel activation via the machine data:

  MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions)

Bit	Value	Meaning
19	0	If bit 0 is also set to "0" in the MD20850 $MC_SPOS_TO_VDI, no auxiliary function M19 is generated in SPOS and SPOSA. This therefore eliminates the acknowledgement time for the auxiliary function.
	1	The implicit auxiliary function M19 is generated and output to the PLC during the programming of SPOS and SPOSA. The address extension corresponds to the spindle number.

Note

Activation via MD35035 should be preferred when using a spindle in multiple channels (axis/spindle exchange).

The auxiliary function M19 is implicitly generated if either of the MD configurations = 1.

After activation, the minimum duration of an SPOS/SPOSA block is increased to the time for output and acknowledgement of the auxiliary functions by the PLC.

The properties of the implicitly generated auxiliary function output M19 are "Quick" and "Output during motion". These properties are fixed settings and are independent of the M19 configuration in the auxiliary function-specific machine data (MD..._$M..._**AUXFU**_...).

There is **no** auxiliary function M19 implicitly generated in the case of spindle positioning instructions via FC 18.

End of positioning

The positioning can be programmed with:

FINEA[S<n>]:	End of motion on reaching "Exact stop fine" (DB31, ... DBX60.7)
COARSEA[S<n>]:	End of motion on reaching "Exact stop coarse" (DB31, ... DBX60.6)
IPOENDA[S<n>]:	End of motion on reaching "IPO stop"

In addition, an end-of-motion criterion for block changes can be set in the braking ramp (100-0%) with IPOBRKA for singleaxis interpolation.

References:
Function Manual, Extended Functions; Positioning Axes (P2)

Block change

The program advances to the next block if the end-of-motion criteria for all spindles or axes programmed in the current block, plus the block change criterion for path interpolation, are fulfilled. This applies to both part-program and technology-cycle blocks.

SPOS, M19 and SPOSA have the same functionality but differ in their block change behavior:

- SPOS and M19

  The block change is carried out if all functions programmed in the block have reached their end-of-block criterion (e.g. all auxiliary functions acknowledged by the PLC, all axes have reached their end points) and the spindle has completed its positioning motion.

- SPOSA

  The program moves to the next block if all the functions (except for spindle) programmed in the current block have reached their end-of-block criterion. If SPOSA is the only entry in the block, block change is performed immediately. The spindle positioning operation may be programmed over several blocks (see WAITS).

Coordination

A coordination of the sequence of motions can be achieved with:

- PLC
- MD configuration
- Programming in the part program

PLC

If the NC/PLC interface signal:
DB31, ... DBX83.5 (spindle in the setpoint range)
is not available, then the channel-specific NC/PLC interface signal:
DB21, ... DBX 6.1 (read-in inhibit)
can be set in order to wait for a spindle to reach a certain position.

MD configuration

Setting:
MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START = 1
is used to perform path interpolation taking the tolerance:
MD35150 $MA_SPIND_DES_VELO_TOL
into account only if the spindle has rotated up to the preselected speed.

The setting
MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START = 2
is used to stop traveling path axes before the start of machining at the last G0.

Machining continues:

- With the next traversing command.
- If the spindle speed is reached.
- When MD35510 $MA_SPIND_STOPPED_AT_IPO_START = 1
  (path feed enable, if spindle stationary).

Programming in the part program

Coordination actions in the part program have the following advantages:

- The part program author can decide at what point in the program the spindle needs to be up to speed, e.g. in order to start machining a workpiece.
- This avoids unnecessary delays.

Coordination in the part program involves programming the `WAITS` instruction:

`WAITS`:	for main spindle (master spindle)
`WAITS[<n>]`:	for spindles with number <n>
`WAITS[<n>,<m>,...]`:	for several spindles up to the maximum number of spindles

⚠ CAUTION

The part program author must ensure that one of the following maintenance conditions occurs for WAITS.

- Position reached
- Spindle stationary
- Spindle up to programmed speed

In cases where one spindle is used in several channels, the part program author must ensure that WAITS starts at the earliest in the phase in which the spindle from another channel has already started to accelerate or decelerate towards the required new speed or direction of rotation.

The control waits before executing subsequent blocks until:

- position(s) programmed with SPOSA are reached.
- spindle standstill is reached with M5:

  DB31, ..., DBX 61.4 (spindle stationary)

  taking into account the tolerance:

  MD36060 $MA_STANDSTILL_VELO_TOL

  WAITS is terminated and the next block loaded when the first occurrence of the signal is detected.
- In M3/M4 (speed control mode), the speed in the setpoint range is:

  DB31, ..., DBX83.5 (spindle in setpoint range)

  taking into account the tolerance:

  MD35150 $MA_SPIND_DES_VELO_TOL

  WAITS is terminated and the next block loaded when the first occurrence of the signal is detected.

  This WAITS function applies in the programmed channel.

  WAITS can be used to wait for all spindles known to this channel, although spindles may also have been started in other channels.

Special cases

- **Tolerance for spindle speed:**

  If the machine data setting is:
  MD35150 $MA_SPIND_DES_VELO_TOL = 0
  the NC/PLC interface signal:
  DB31, ... DBX83.5 (spindle in setpoint range)
  is always set to 1.

  WAITS is terminated as soon as the spindle has reached the setpoint-side target after a change in speed or direction (M3/M4).

- **Missing enable signals:**

  If the WAITS function waits for the "Spindle in setpoint range" signal in speed control mode and the spindle stops or fails to rotate because an enable signal (axial feed enable, controller, pulse enable, etc.) is missing, the block is not terminated until the "Spindle in setpoint range" signal is active, once enable signals are being received again.

- **Response to NC and mode-group stop:**

  If an NC or mode-group stop is triggered during WAITS, the wait operation is resumed after the NC start with all the above conditions.

Note

In particular when using spindles across different channels, care should be taken when programming not to start WAITS too early in one channel, i.e. at a time when the spindle in the other channel is still rotating at its "old" speed.
In such cases, the "Spindle in setpoint range" signal is activated and WAITS is stopped too soon.
To prevent this happening, it is strongly recommended to set a WAITM before WAITS.

Feedrate

The positioning speed is configured in the machine data:

MD35300 $MA_SPIND_POSCTRL_VELO (position control switching speed)

The configured positioning speed can be modified by programming or by synchronized actions:
`FA[S<n>]=<value>`

where: `<n>`: Spindle number
`<value>`: Positioning speed in degrees/min

With `FA[S<n>]=0`, the configured speed takes effect.

Acceleration

The accelerations are configured in the machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL(acceleration in position control mode)

MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL(Acceleration in the speed control mode)

The configured dynamic response during positioning can be modified by programming or by synchronized actions:
`ACC[S<n>]=<value>`

where: `<n>`: Spindle number
`<value>`: Acceleration as a percentage of the configured acceleration

With `ACC[S<n>]=0`, the configured acceleration takes effect.

Aborting the positioning process

The positioning action is aborted:

- By the NC/PLC interface signal:

  DB31, ... DBX2.2 (delete distance-to-go / spindle reset).
- With every reset (e.g. operator panel front reset).
- Through NC stop.

The abort response in independent of the machine data:

MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (individual spindle reset)

Special features

The spindle override switch is active.

15.2.5.2 Positioning from rotation

Initial situation

The spindle can be in speed control mode or in position control mode when positioning starts (`SPOS`, `M19` or `SPOSA` command in the program).

One must distinguish between the following cases:

Case 1:	Spindle in speed control mode, encoder limit frequency exceeded
Case 2:	Spindle in speed control mode, encoder limit frequency not exceeded
Case 3:	Spindle in position control mode
Case 4:	Spindle speed ‹ Position-control activation speed

Procedure

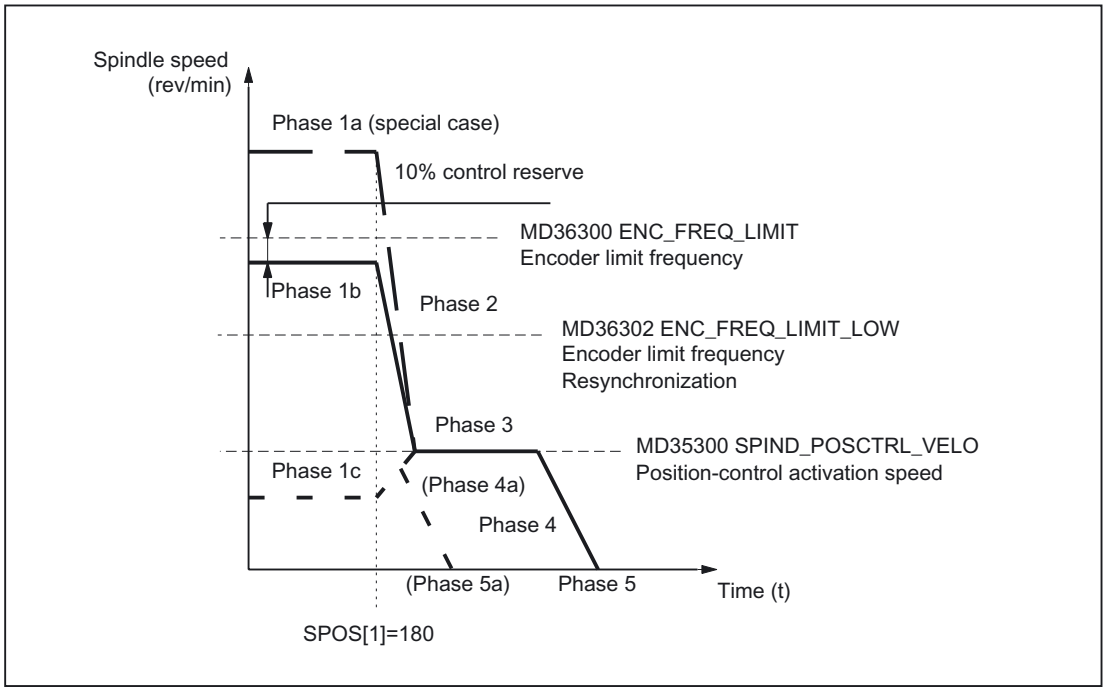


Figure 15-1 Positioning from rotation

Note

The speed arising from the configuration of the encoder limit frequency for the resynchronization of the encoder (MD36302 $MA_ENC_FREQ_LOW) must be greater than the position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VELO).

Phase 1

Positioning from phase 1a:

The spindle is rotating at a higher speed than the encoder limit frequency. The spindle is not synchronized.

Positioning from phase 1b:

The spindle is rotating at a lower speed than the encoder limit frequency. The spindle is synchronized.

Note

If the position control is active, the speed can only amount to 90% of the maximum speed of the spindle or the encoder limit frequency (10% control reserve required).

Positioning from phase 1c:

The spindle rotates at the programmed spindle speed whereby the speed is lower than the configured position-control activation speed:

MD35300 $MA_SPIND_POSCTRL_VELO

The spindle is synchronized.

Phase 2

Spindle speed > Position-control activation speed

When the SPOS, M19 or SPOSA command is activated, the spindle begins to slow down to the position-control activation speed with the configured acceleration:

MD35200 $MA_GEAR_STEP_SPEEDCTL_ACCEL

The spindle is synchronized once the encoder limit frequency threshold is crossed.

Spindle speed < Position-control activation speed

SPOS, M19 or SPOSA are programmed to switch the spindle to position control mode (if it is not already in that mode).

The configured acceleration in position control mode is activated:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

The travel path to the target point is calculated.

The spindle travels to the programmed end point optimally in terms of time. This means that the end point is approached at the highest possible speed (maximum MD35300 $MA_SPIND_POSCTRL_VELO). Depending on the appropriate secondary conditions, phases 2 - 3 - 4 - 5 or 4a - 5a are executed.

Phase 3

Spindle speed > Position-control activation speed

When the configured position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VELO) is reached:

- Position control is activated (if not already active).
- The distancetogo (to the target point) is calculated.
- There is a switch to the configured configured acceleration in position control mode (or this acceleration is retained):

  MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Spindle speed < Position-control activation speed

The spindle was accelerated up to the configured position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VELO) to reach the end point. This is not exceeded.

The braking start point calculation detects when the programmed spindle position can be approached accurately at the acceleration configured in position control mode (MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL).

Phase 4

Spindle speed > Position-control activation speed

The spindle brakes from the calculated "braking point" with machine data:
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL
to the target position.

Spindle speed < Position-control activation speed

At the time identified by the braking start point calculation in phase 3, the spindle brakes to a standstill with the acceleration configured in position control mode (MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL).

Phase 4a:

When the SPOS command is activated the proximity of the end point is such that the spindle can no longer be accelerated to the configured position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VELO).

The spindle brakes to a standstill with the acceleration configured in position control mode (MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL).

Phase 5

Spindle speed > Position-control activation speed

Position control remains active and holds the spindle in the programmed position.

Note

The maximum encoder limit frequency of the spindle position actual-value encoder is monitored by the control (it may be exceeded); in position control mode, the setpoint speed is reduced to 90% of the measuring system limit speed.

The following NC/PLC interface signal is set:

DB31, ... DBX83.1 (programmed speed too high)

If "MS limit frequency exceeded" is still pending following a reduction in the setpoint speed, an alarm is output.

Spindle speed < Position-control activation speed (Phase 5, 5a)

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

If the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the configured exact stop fine and coarse limits, the following NC/PLC interface signals are set:

DB31, ... DBX60.7 (Position reached with coarse exact stop)

DB31, ... DBX60.7 (Position reached with fine exact stop)

The exact stop limits are defined with the machine data:

MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)

MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)

Note

The positioning procedure is considered complete when the end-of-positioning criterion is reached and signaled.

The condition is "Exact stop fine". This applies to `SPOS`, `M19` or `SPOSA` from the part program, synchronized actions and spindle positioning by the PLC using FC 18.

15.2.5.3 Positioning from standstill

Procedure

A distinction is made between two cases with regard to positioning from standstill:

- Case 1: The spindle is not synchronized.

  This is the case if the spindle is to be positioned after switching on the control and drive or after a gear step change (e.g. for a tool change).

  MD31040 $MA_ENC_IS_DIRECT = 0

- Case 2: The spindle is synchronized.

  This is the case if, after switching on the control and drive, the spindle is to be rotated through a minimum of one revolution with M3 or M4 and then stopped with M5 (synchronization with the zero mark) before the first positioning action.

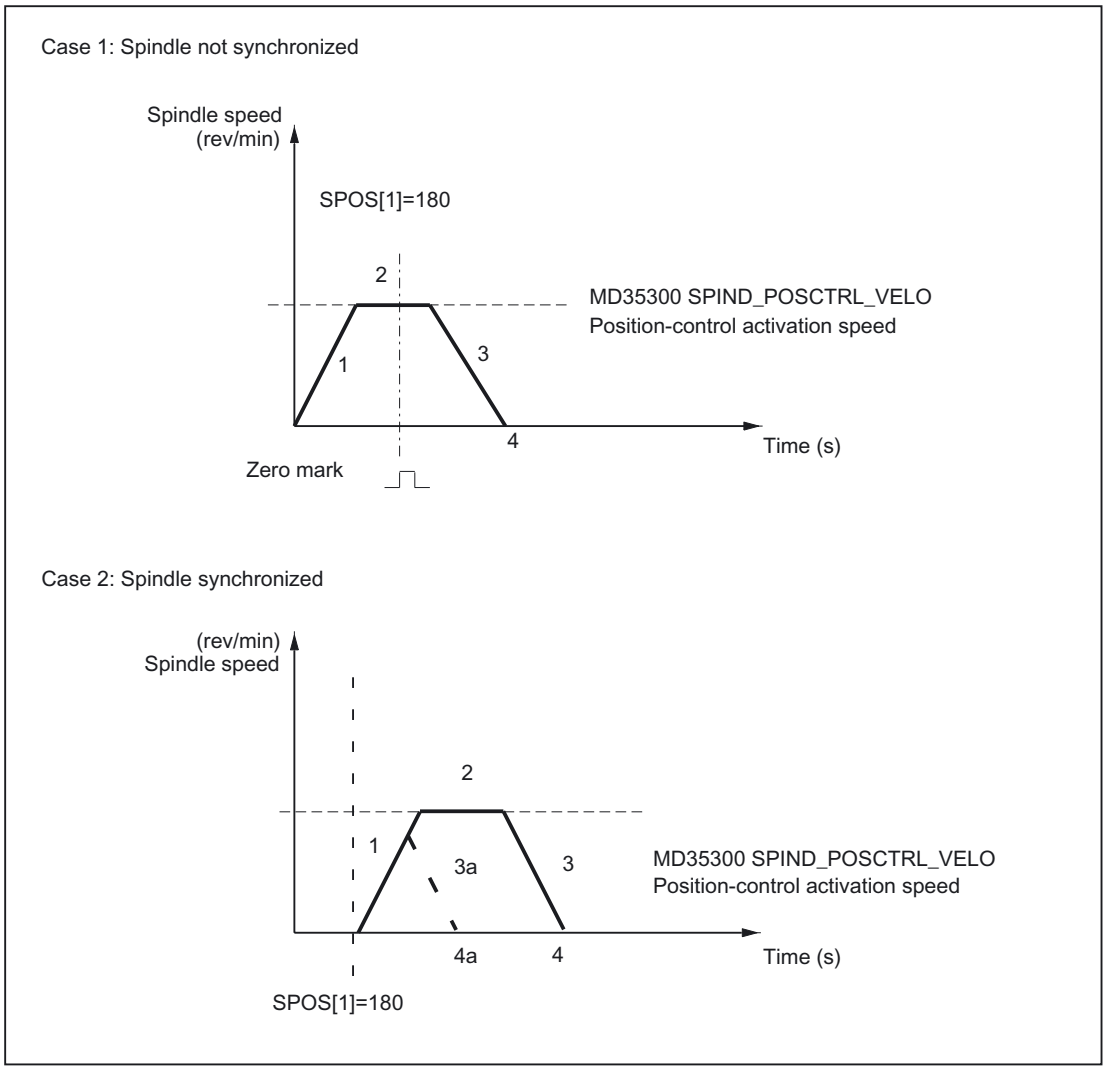


Figure 15-2 Positioning with stationary spindle

Phase 1

Case 1: Spindle not synchronized

With the programming of SPOS, M19 or SPOSA the spindle accelerates with the acceleration from the machine data:

MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL (Acceleration in the speed control mode)

This direction of rotation is defined by the machine data:

MD35350 $MA_SPIND_POSITIONING_DIR (Direction of rotation while positioning to standstill)

Exception:

If ACN, ACP, IC is used for positioning, the programmed direction of travel is activated.

The spindle is synchronized at the next zero mark of the spindle position actual-value encoder and switches to the position control mode.

Whether the zero mark is found in the traversed path (except for IC), is monitored:

MD34060 $MA_REFP_MAX_MARKER_DIST (maximum distance to the reference mark)

If the speed defined in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO (Positioning speed)
is reached before the spindle is synchronized, the spindle will continue to rotate at the positioning activation speed (it is not accelerated further).

Case 2: Spindle synchronized

SPOS, M19 or SPOSA will switch the spindle to position control mode.

The acceleration from the following machine data is active:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL (acceleration in position control mode)

The direction of rotation is defined by the programmed motion (ACP, ACN, IC, DC) or via the pending distance-to-go.

The speed entered in
MD35300 $MA_SPIND_POSCTRL_VELO (position control activation speed)
is not exceeded in the machine data.

The travel path to the end position is calculated.

The spindle travels to the programmed end point optimally in terms of time. This means that the end point is approached at the highest possible speed (maximum MD35300 $MA_SPIND_POSCTRL_VELO). Depending on the appropriate secondary conditions, the phases 1 - 2 - 3 - 4 or 1- 3a - 4a are executed.

Phase 2

Case 1: Spindle not synchronized

When the spindle is synchronized, position control is activated.

The spindle rotates at the maximum speed stored in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO
until the braking start point calculation identifies the point at which the programmed spindle position can be approached accurately with the defined acceleration.

Case 2: Spindle synchronized

To reach the end point, the spindle is accelerated up to the speed defined in machine data:
MD35300 $MA_SPIND_POSCTRL_VELO.

This is not exceeded.

The braking start point calculation identifies when the programmed spindle position can be approached accurately at the acceleration defined in machine data:
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL.

At the point, which is determined by the braking start point calculation in Phase 1, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Phase 3

At the point, which is determined by the braking start point calculation in Phase 2, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Phase 3a:

When the SPOS command is activated the proximity of the end point is such that the spindle can no longer be accelerated up to machine data:
MD35300 $MA_SPIND_POSCTRL_VELO.

The spindle is braked to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

Phase 4, 4a

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

NC/PLC IS:
DB31, ... DBX60.6 (Position reached with exact stop coarse)
and
DB31, ... DBX60.7 (Position reached with exact stop fine)
are set if the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the settings for the exact stop fine and coarse limits.

This is defined in the machine data:

MD36010 $MA_STOP_LIMIT_FINE

MD36000 $MA_STOP_LIMIT_COARSE

Phase 3:

At the point, which is determined by the braking start point calculation in Phase 2, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_ POSCTRL_ACCEL

Phase 4:

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

NC/PLC IS:
DB31, ... DBX60.6 (Position reached with exact stop coarse)
and
DB31, ... DBX60.7 (Position reached with exact stop fine)
are set if the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the settings for the exact stop fine and coarse limits.

This is defined in the machine data:

MD36010 $MA_STOP_LIMIT_FINE

MD36000 $MA_STOP_LIMIT_COARSE

15.2.5.4 "Spindle in position" signal for tool change

Function

The motion sequence for a tool change, especially for milling machines, mainly comprises positioning the spindle and then the subsequent (for optimization runs, also at the same time) approach to the tool change position with the path axes. In this case, the mandatory requirement is that the spindle is reached before approaching the tool change position.

If the tool change cycle is interrupted by the machine operator (e.g. with an NC stop, NC stop axes plus spindles, mode group stop, etc.), then it must be completely ruled out that the spindle moves into the tool changer at an incorrect position.

This is the reason that for spindle positioning, when the last programmed spindle position is reached with "Exact stop fine" the following NC/PLC interface signal is output to check the position:

DB31, ... DBX85.5 (spindle in position)

Note

The signal is only output for the "Spindle positioning" function.

This includes:

- SPOS, SPOSA and M19 in the part program
- SPOS and M19 in synchronized actions
- Spindle positioning, using FC18
- Spindle positioning via PLC interface (DB31, ... DBX30.4)

Setting the signal

Requirements for output of signal DB31, ... DBX85.5 (spindle in position) are as follows:

- The reference state of the spindle:

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

  Note

  When positioning the spindle, the zero mark is automatically searched for. This is the reason that for an error-free sequence, the referenced signal is always available at the end of positioning movement.

- "Exact stop fine" must have been reached:

  DB31, ... DBX60.7 (exact stop fine) = 1

  Additionally, the last programmed spindle position must have been reached on the setpoint side.

Deleting the signal

When signal DB31, ... DBX60.7 is withdrawn (exact stop fine), then signal DB31, ... DBX85.5 (spindle in position) is also always reset.

Additional properties

- If the spindle is already at the programmed position after a positioning, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) remains set.
- If, after a positioning ("Spindle in position" signal was output) the spindle is traversed, e.g. in the JOG mode, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) is deleted.

  If the spindle is returned to its original position in this mode, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) is set again. The last position selection is kept.

15.2.6 Axis mode

15.2.6.1 General functionality

Functionality

If for certain machining tasks, e.g. on lathes with end-face machining, it is not sufficient to traverse the spindle exclusively under speed control via M3, M4, M5 or to position with SPOS, M19 or SPOSA, the spindle can be switched to position-controlled axis mode and traversed as a rotary axis.

Examples of rotary axis functions:

- Programming with axis name
- Zero offsets (G54, G55, TRANS, etc.)
- G90, G91, IC, AC, DC, ACP, ACN
- Kinematic transformations (e.g. TRANSMIT)
- Path interpolation
- Traversing as positioning axis

References:
Function Manual, Extended Functions; Section "Rotary axes (R2)"

Preconditions

- The same spindle motor is used for spindle mode and axis mode.
- The same position measurement system or separate position measurement systems can be used for spindle mode and axis mode.
- An actual position value encoder is a mandatory requirement for axis mode.
- The spindle must be referenced for use of the axis mode, e.g. referenced with `G74`.

  Example:

Program code	Comment
`M70`	`; Switch spindle over to axis mode`
`G74 C1=0 Z100`	`; Reference axis`
`G0 C180 X50`	`; Traverse axis position-controlled`

Configurable M function

The M function used to switch the spindle to axis mode can be configured channel-specifically via the following machine data:

MD20094 $MC_SPIND_RIGID_TAPPING_M_NR

Note

The controller detects the transition to axis mode automatically from the program sequence (see "Implicit transition to axis mode (Page 1411)"). The explicit programming of the configured M function for switching the spindle to axis mode in the part program is therefore not necessary. However, the M function can continue to be programmed, e.g. to increase the readability of the part program.

Special features

- The feed override switch is active.
- The NC/PLC interface signal does not terminate the axis mode per default:
  DB21, ... DBX7.7 (reset).
- The NC/PLC interface signals:
  DB31, ... DBB16 to DBB19 and DBB82 to DBB91
  are not important if:
  DB31, ... DBX60.0 (axis / no spindle) = 0
- Axis mode can be activated in all gear steps.

  If the actual position value encoder is installed on the motor (indirect measuring system), the positioning and contouring accuracy can vary for the different gear steps.
- The gear step cannot be changed when the axis mode is active.
  The spindle must be switched to control mode.
  This is done with `M41` ... `M45` and `M5`, `SPCOF`.
- In axis mode, the first parameter set is active (machine data index = zero).
  References
  Function Manual, Basic Functions; Section "Velocities, setpoint / actual value systems, closed-loop control (G2)" > "Closed-loop control" > "Parameter sets of the position controller"

Dynamic response

The dynamic limits of the axis apply in axis mode. For example:

- MD32000 $MA_MAX_AX_VELO[<axis>] (maximum axis velocity)
- MD32300 $MA_MAX_AX_ACCEL[<axis>] (maximum axis acceleration)
- MD32431 $MA_MAX_AX_JERK[<axis>] (maximum axial jerk for path motion)

Feedforward control

The feedforward control mode active for the axis is retained.

A detailed description of the "Dynamic feedforward control" function can be found in:

References
Function Manual, Extended Functions; Section "Compensations (K3)" > "Dynamic feedforward control (following error compensation)"

Example: Resolution switchover for analog actuator

Switching to axis mode

```
Programming                Comment
SPOS=...
M5                         ; Controller enable off (by PLC)
                             → is output on PLC
M70                        ; Switch actuator (by PLC on account of M70)
                             Controller enable on (by PLC)
C=...                      ; NC traverses with axis parameter set
```

Switching to spindle mode

```
Programming                Comment
C=...
M71                        ; → Output to PLC
                             Closed-loop controller enable off (by PLC)
                             Switch actuator (by PLC)
                             Switched to spindle parameter set (1-5) internally in
                             the NC, controller enable on (by PLC)
M3/4/5 or SPOS=...         ; NC traverses with spindle parameter set
```

Change to spindle mode

The appropriate parameter set 1 ... 5 is selected for the active gear stage.

Except for tapping with compensating chuck, the feedforward control is switched on when the following applies:
MD32620 $MA_FFW_MODE (feedforward control type) ≠ 0

Parameter set	Axis mode	Spindle mode
1	Valid	-
2	-	Valid
3	-	Valid
4	-	Valid
5	-	Valid
6	-	Valid
Spindle mode: Parameter set according to the gear stage		

15.2.6.2 Implicit transition to axis mode

Function

The control system detects the transition to axis mode automatically from the program sequence and generates the requisite M70 sequence within the control system. The situation will dictate which steps are performed. At most, these will include:

1. Stopping the spindle
2. Switching on of the position control, treatment of feedforward control, and parameter block changeover
3. Position synchronization of the block preparation (internal preprocessing stop, if necessary)

This function is always active. Explicit programming of M70 in the part program is, therefore, essentially not necessary.

Procedure

Sequence of the implicit transition to axis mode (M70 is not programmed in the part program):

- Transition from speed control mode (M3, M4, M5, SPCOF, ...) to axis mode:

  The transition is detected internally by the control, and an intermediate block is inserted in front of the block which requests the axis mode. The block created contains the M70 functionality. The execution duration for this block is more or less the same as the time required to execute a programmed M70 block. Differences may arise in the event of short switchovers when the spindle is stationary (no braking time) if the implicit generation and output of the auxiliary function M70 to the PLC is dispensed with (see MD35035).

- Transition from positioning mode (M19, SPOS, SPOSA) to axis mode:

  The transition is executed immediately and without the generation of an intermediate block. Configured accordingly (see MD35035), the auxiliary function M70, which is generated implicitly, is output to the PLC when the block in which the spindle has its axis mode is loaded.

Output of auxiliary functions to PLC

The implicit transition to axis mode can be notified to the PLC in the form of an auxiliary function output.

Activation/Deactivation

The activation/deactivation of this functionality is done using machine data:

MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions)

Bit	Value	Meaning
20	0	No auxiliary function output to the PLC in the case of M70 functionality which is generated inside the control.
	1	In the case of M70 functionality which is generated inside the control, the auxiliary function M70 is generated and output to the PLC. The address extension corresponds to the spindle number.

Note

An auxiliary function M70 which is programmed in the part program is always output to the PLC.

Properties

The properties of the implicitly generated auxiliary function output M70 are "Quick" and "Output during motion". These properties are fixed settings and are independent of the M70 configuration in the auxiliary-function-specific machine data (MD..._$M..._**AUXFU**_...).

M70 is only generated once during transition to axis mode. No further M70 auxiliary functions are generated and output in adjacent blocks in which the spindle is operated as an axis. M70 is not implicitly generated and output again until axis mode is exited via, for example, SPOS, M3, M4, M5, SPCOF, etc. and following a renewed transition to axis mode.

Constraints

Synchronized actions

When the spindle is programmed as an axis in synchronized actions, it is essential to continue making provisions in the application to ensure there are criteria for the transition to axis mode.

If the spindle is in speed control mode, the instruction `M70` or `SPOS` must be programmed prior to programming as an axis. Otherwise alarm signals occur during axis programming.

FC 18

As with synchronized actions, transition to axis mode must also be undertaken on the application side in FC 18, e.g. through preparatory positioning instructions. Otherwise, the FC 18 call is acknowledged with an error bit in the FC 18 status word.

No auxiliary function M70 is implicitly generated in the event of transition to axis mode through programming via FC 18.

Examples

Example 1:

Part program: Transition from rotating spindle to axis mode

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

```
Program code            Comment
N05 M3 S1000
N10 ...
N15 POS[C]=77           ; Before loading N15, an M70 intermediate block is generated
                          in which the spindle is stopped, and M70 is output to the
                          PLC.
…
```

Example 2:

Part program: Transition from positioning mode to axis mode

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

```
Program code            Comment
N05 SPOS=0
N10 ...
N15 C77                 ; Output of the implicit M70 to the PLC, no intermediate
                          block.
…
```

Example 3:

Synchronized actions: Transition from spindle positioning mode to axis mode

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

```
Program code                              Comment
WHEN COND1==TRUE DO SPOS=180
WHEN COND2==TRUE DO POS[C]=270            ; Output of the implicit M70 to the PLC.
```

Example 4:

Synchronized actions: Transition from speed control mode to axis mode with M70

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

```
Program code                              Comment
WHEN COND11==TRUE DO M3 S1000
WHEN COND12==TRUE DO M70                  ; Output of M70 to the PLC.
WHEN COND13==TRUE DO POS[C]=270           ; No generation of an implicit M70 because
                                            axis mode already exists.
```

Example 5:

Synchronized actions: Invalid transition from speed control mode to axis mode

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

```
Program code                              Comment
WHEN COND21==TRUE DO M3 S1000
WHEN COND22==TRUE DO POS[C]=270           ; Alarm 20141!
```

15.2.7 Initial spindle state

Spindle basic setting

The following machine data is used to specify a spindle mode as basic setting:

MD35020 $MA_SPIND_DEFAULT_MODE

Value	Spindle basic setting
0	Speed control mode, position control deselected
1	Speed control mode, position control activated
2	Positioning mode
3	Axis mode

Time when the spindle basic setting takes effect

The time when the spindle basic setting takes effect is set in the machine data:

MD35030 $MA_SPIND_DEFAULT_ACT_MASK

Value	Effective time
0	POWER ON
1	POWER ON and program start
2	POWER ON and RESET (M2 / M30)

15.3 Reference / synchronize

Why synchronize?

In order to ensure that the controller detects the exact position of the spindle when it is switched on, the controller must be synchronized with the position measuring system of the spindle.

The following functions are possible only with a synchronized spindle:

- Thread cutting
- Tapping without compensating chuck
- Axis programming

For further explanations about synchronization of the spindle, see Section "R1: Referencing (Page 1319)".

Why reference?

In order to ensure that the controller detects the exact machine zero when it is switched on, the controller must be synchronized with the position measurement system of the rotary axis. This process is known as referencing. The sequence of operations required to reference an axis is known as search for reference.

Only a referenced axis can approach a programmed position accurately on the machine.

For further explanations about referencing the rotary axis, see Section "R1: Referencing (Page 1319)".

Installation position of the position measurement system

The position measurement systems can be installed as follows:

- Directly on the motor in combination with a Bero proximity switch on the spindle as a zero-mark encoder
- On the motor via a measuring gearbox in combination with a Bero proximity switch on the spindle as a zero-mark encoder
- Directly on the spindle
- On the spindle via a measuring gearbox in combination with a Bero proximity switch on the spindle as a zero-mark encoder (only with ratios not equal to 1:1)

Where two position measuring systems are provided, they can be installed either in the same location or separately.

Synchronization procedure

When the spindle is switched on, it can be synchronized as follows:

- The spindle is started with a spindle speed (S value) and a spindle rotation (M3 or M4) and synchronized with the next zero mark of the position measurement system or with the next Bero signal.
- The spindle is to be positioned from standstill using SPOS, M19 or SPOSA. The spindle synchronizes with the next zero mark of the position measurement system or with the next Bero signal. It is then positioned to the programmed position.
- The spindle can be synchronized from the motion (after M3 or M4) using SPOS, M19 or SPOSA.

  The responses are as follows:

  – With SPOS=<Pos>, SPOS=DC(<Pos>) and SPOS=AC(<Pos>), the direction of motion is retained and the position is approached.

  – With SPOS = ACN(<Pos>) or SPOS = ACP(<Pos>), the position is always approached with a negative or positive direction of motion. If necessary, the direction of motion is inverted prior to positioning.
- Crossing the zero mark in JOG mode by means of direction keys in speed control mode.

Note

It does not make any difference whether the synchronization procedure is initiated from the part program, FC 18 or synchronized actions.

Note

During synchronization of the spindle, all four possible reference point values are effective depending on the measuring system selected. The measurement system offset has the same effect.

The following machine data must be observed:

- MD34080 $MA_REFP_MOVE_DIST
  (Reference point distance / destination point for a distance-coded system)
- MD34090 $MA_REFP_MOVE_DIST_CORR
  (Reference point offset / absolute offset, distance-coded)
- MD34100 $MA_REFP_SET_POS
  (Reference point value, with distance-coded system without any significance)

If a non-referenced spindle with SPOS=IC(...) and a path < 360 degrees is positioned, it may be the case that the zero mark is not crossed and the spindle position is still not synchronized with the zero mark. This can happen:

- After POWER ON
- By setting the axial NC/PLC interface signals:
  DB31, ... DBX17.5 (resynchronize spindle when positioning 2)
  DB31, ... DBX17.4 (resynchronize spindle when positioning 1)

Special features for synchronization with BERO

The position falsification caused by the signal delay with BERO can be corrected internally in the NC by entering a signal runtime compensation.

The signal runtime compensation is set by means of the machine data:

- MD31122 $MA_BERO_DELAY_TIME_PLUS

  (BERO delay time for a positive direction of motion)

- MD31123 $MA_BERO_DELAY_TIME_MINUS

  (BERO delay time for a negative direction of motion)

The effect depends on the setting in machine data:

MD34200 $MA_ENC_REFP_MODE (referencing mode)

MD34200 = 7	The setting MD34200 $MA_ENC_REFP_MODE = 7 only executes position synchronization at a velocity/speed which is fixed in machine data: MD34040 $MA_REFP_VELO_SEARCH_MARKER (reduced velocity) The zero mark is not automatically sought, it has to be requested explicitly with the 0-1 edge of the NC/PLC interface signal: DB31, ... DBX16.4/5 (resynchronize spindle 1/2*) *) 1/2 stands for the selected measuring system. The velocity defined in MD34040 is also effective when referencing in JOG-REF mode and through the part program with G74.
MD34200 = 2	Setting MD34200 $MA_ENC_REFP_MODE = 2 executes position synchronization without specifying a specific velocity/speed.

Note

Signal propagation delays are preset on delivery so that the content generally does not have to be changed.

Referencing sequence

If the spindle is to be programmed in axis mode directly after controller power-up, it must be ensured that the axis is referenced.

When the controller is switched on, the spindle can be referenced (condition is one zero mark per revolution).

For information about the referencing procedure, see Section "R1: Referencing (Page 1319)".

The rotary axis is referenced at the same time as the spindle is synchronized (see section "Synchronization procedure") if the position measuring system used for the spindle is also used for the rotary axis.

Position measurement systems, spindle

The spindle can be switched from spindle mode to axis mode (rotary axis) if a single motor is used for spindle mode and axis mode.

The spindle (spindle mode and axis mode) can be equipped with one or two position measurement systems. With two position measurement systems, it is possible to assign one position measurement system to the spindle and the other to the rotary axis, or to assign two position measurement systems to the spindle. Where two position measurement systems are provided, both are updated by the controller, but only one can be active.

The active position measuring system is selected using the NC/PLC interface signal:
DB31, ... DBX1.5 (position measuring system 1)
or
DB31, ... DBX1.6 (position measuring system 2)

The active position measurement system is required for the following functions:

- Position control of the spindle (SPCON)
- Spindle positioning (SPOS, M19 and SPOSA)
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck (G331, G332)
- Revolutional feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- Spindle actual speed display
- Axis mode
- Synchronous spindle setpoint coupling

Resynchronizing the position measuring system for the spindle

In the following cases, the spindle position measurement system must be resynchronized:

- The position encoder is on the motor, a Bero proximity switch is mounted on the spindle and a gear stage change is performed. Synchronization is triggered internally once the spindle is rotating in the new gear stage (see Synchronization procedure).
- The machine has a selector switch for a vertical and horizontal spindle. Two different position encoders are used (one for the vertical spindle and one for the horizontal spindle), but only one actual value input is used on the controller. When the system switches from the vertical to the horizontal spindle, the spindle must be resynchronized.

  This synchronization is initiated with the NC/PLC interface signal:

  DB31, ... DBX16.4 (resynchronize spindle 1)
  or
  DB31, ... DBX16.5 (resynchronize spindle 2)

  The spindle must be in open-loop control mode.

Position restoration with POWER ON

For spindles with incremental position measuring systems, it is possible to buffer the actual values after a POWER OFF and after POWER ON, to restore the position last buffered before switching-off, in order that position-dependent functions, e.g. transformation can be restored (see Section "Auto-Hotspot"). One application is, e.g. tool retraction after POWER OFF when machining with tool orientation (see Section "Tool withdrawal after POWER ON with orientation transformation (Page 663)").

The following NC/PLC interface signals display the state of the position measuring system after position restoration:

DB31, ... DBX71.4 ("Restored 1") for position measuring system 1

DB31, ... DBX71.5 ("Restored 2") for position measuring system 2

Once the tool has been retracted in the JOG mode, axes whose positions have been restored are referenced. As a consequence, signals DB31, ... DBX71.4/5 ("Restored 1/2") are deleted and signals DB31, ... DBX60.4/5 ("Referenced/synchronized 1/2") are set.

Note

If machine data MD20700 $MC_REFP_NC_START_LOCK is set to a value of "2", then an NC start is also possible with "restored" axis positions (in the MDA mode or when overstoring).

15.4 Configurable gear adaptation

15.4.1 Gear stages for spindles and gear change change

Why do we need gear stages?

Gear stages are used on spindles to step down the speed of the motor in order to generate a high torque at low spindle speeds or to step up in order to maintain a high speed.

No. of gear stages

Five gear stages can be configured for each spindle.

The number of used gear stages is defined in machine data:

MD35090 $MA_NUM_GEAR_STEPS

Parameterization of the gear stages

The gear stages 1 to 5 can be parameterized via the following machine data:

Machine data	Meaning
MD35012 $MA_GEAR_STEP_CHANGE_POSITION[<n>]	Gear stage change position
MD35110 $MA_GEAR_STEP_MAX_VELO[<n>]	Maximum speed for automatic gear stage change
MD35120 $MA_GEAR_STEP_MIN_VELO[<n>]	Minimum speed for automatic gear stage change
MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]	Maximum speed of gear stage
MD35135 $MA_GEAR_STEP_PC_MAX_VELO_LIMIT[<n>]	Maximum speed of gear stage in position control
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[<n>]	Minimum speed of gear stage
MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL[<n>]	Acceleration in speed control mode
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL[<n>]	Acceleration in position control mode
MD35300 $MA_SPIND_POSCTRL_VELO[<n>]	Position control activation speed
MD35310 $MA_SPIND_POSIT_DELAY_TIME[<n>]	Positioning delay time
MD35550 $MA_DRILL_VELO_LIMIT[<n>]	Maximum speed for tapping without compensating chuck

Type of gear stage change

The type of gear stage change is set in machine data:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE

Bit	Value	Meaning
0	0	The spindle motor is attached to the spindle directly (1:1) or with a non-variable transmission ratio (basic setting). The machine data of the first gear stage is effective.
	1	Spindle motor with up to five gear stages. The gear stage change takes place: • In oscillation mode • At indefinite change position
1	0	Meaning as in Bit 0 = 0.
	1	Meaning as in Bit 0 = 1, however, the gear stage change takes place at the configured spindle position. The change position is set in machine data: MD35012 $MA_GEAR_STEP_CHANGE_POSITION The position is approached in the current gear stage before the gear stage change. If Bit 1 is set, then Bit 0 is ignored!
3	1	The gear stage change dialog between NCK and PLC is simulated.
5	1	The second gear stage data set is used while tapping with G331/G332 (see the following paragraph "Second gear stage data set"). The bit must be set for the master spindle used during the tapping.

Requirement for a gear stage change

In principle, the gear stage change is only performed if the requested gear stage is not the same as the active gear stage.

Parameter set selection during gear stage change

The servo parameter set is also changed over with the gear stage if:

MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0 or 1

For further information, see Section "Parameter set selection during gear step change (Page 1437)".

Request gear stage change

A gear stage change can be requested:

- In the part program using:
  - `M40 S...`

    Automatic gear stage selection to the programmed speed `S...`
  - `M41 ... M45`

    Direct selection of gear stages 1 ... 5
  - `M70`

    For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5

    (see "Configurable gear step in M70 (Page 1456)")
  - `G331 S...`

    For MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, Bit 5 = 1
- In synchronized actions using:
  - `DO M40 S...`

    Automatic gear stage selection to the programmed speed `S...`
  - `DO M41... M45`

    Direct selection of gear stages 1 ... 5
  - `DO M70`

    For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5
- Through the PLC using the FC18 function block
- In the reset state through description of NC/PLC interface:

  DB31, ... DBX16.0-16.2 (actual gear stage A to C)

  The mechanically active gear stage can be communicated to the NC especially after a POWER ON.

Note

If the spindle motor is attached to the spindle directly (1:1) or with a non-variable transmission ratio (MD35010 = 0), then the M40 and M41 ... M45 auxiliary functions are not relevant to this spindle.

Gear stage change

Gear stage selection between two gear stages with specification of a maximum spindle speed is shown in the example below:

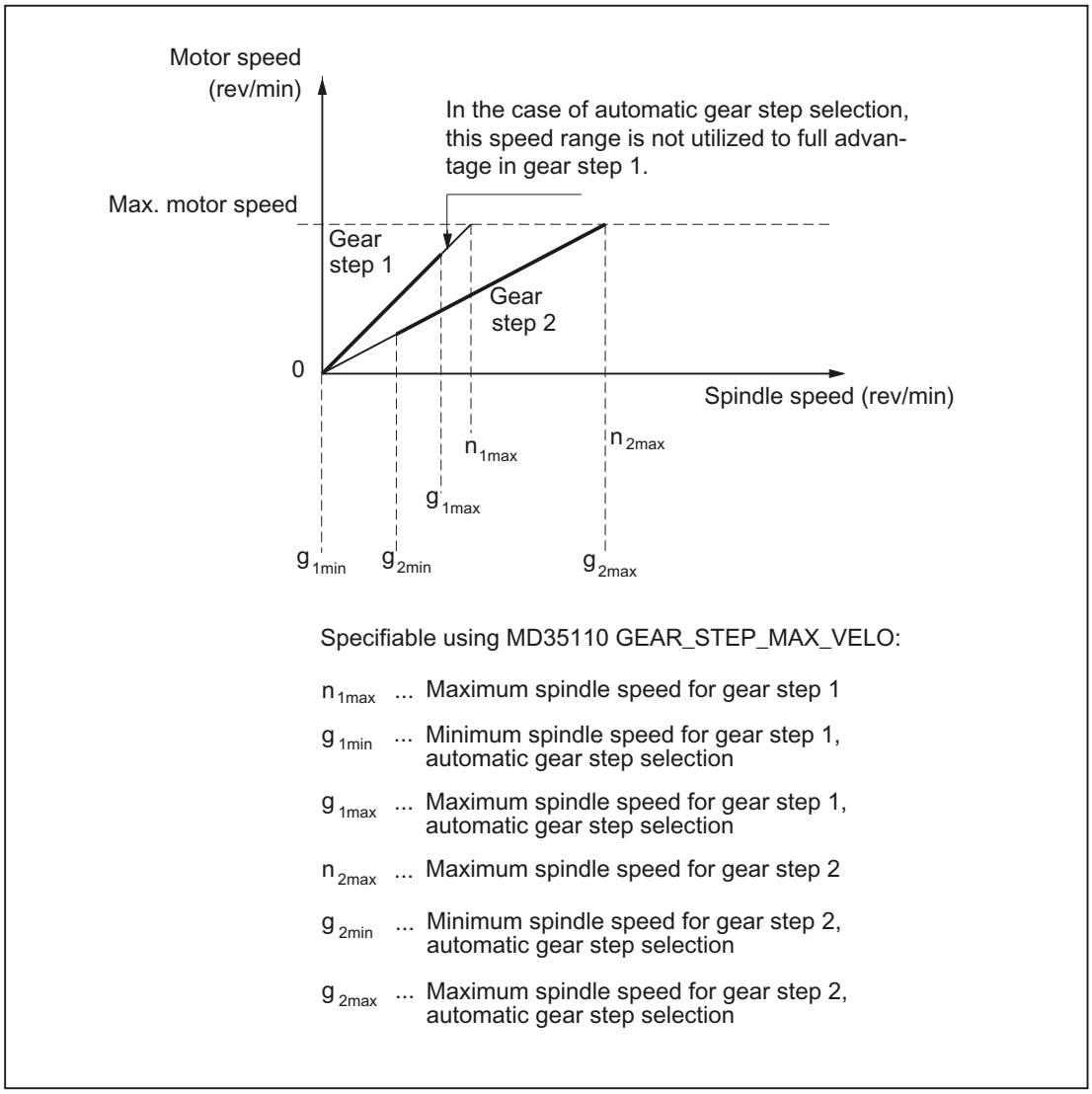


Figure 15-3 Gear stage change with selection between two gear stages

Process sequence of the gear stage change

If the new gear stage is preselected, the following sequence is implemented:

1. Changeover sequence

   The two following NC/PLC interface signals are set:

   DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)

   DB31, ... DBX82.3 (change over gear stage)

   In accordance with the point at which NC/PLC IS:
   DB31, ... DBX18.5 (oscillation speed)
   is set, the spindle decelerates to a standstill at the acceleration for oscillation or at the acceleration for speed control / position control.

   Oscillation can be activated at the latest when the spindle reaches a standstill:
   DB31, ... DBX61.4 (axis/spindle stationary)
   with NC/PLC IS:
   DB31, ... DBX18.5 (oscillation speed).

   In principle, the new gear stage can also be engaged without oscillation

   When the new gear stage is engaged, the following NC/PLC interface signals are set by the PLC program:

   DB31, ... DBX16.0-16.2 (actual gear stage A to C)

   DB31, ... DBX16.3 (gear is changed)

2. End of gear stage change

   The gear stage change is treated as completed (spindle operation type "oscillation mode" is deselected), if the following NC/PLC interface signal is set:

   DB31, ... DBX16.3 (gear is changed)

   The new actual gear stage is changed to the servo and interpolation parameter set when the motor is stationary.

   With NC/PLC interface signal:
   DB31, ... DBX16.3 (gear is changed)
   is used to communicate to the NC that the new gear stage is valid and the oscillation mode can be completed.

   NC/PLC IS:
   DB31, ... DBX82.3 (change gear)
   is reset by the NCK,
   which causes the PLC program to reset NC/PLC IS:
   DB31, ... DBX16.3 (gear changed).

In this case, it does not matter whether NC/PLC IS:
DB31, ... DBX18.5 (oscillation mode)
is still set.

The actual gear stage, which should correspond to the set gear stage, is relevant for selecting the parameter set.

If this is not the case, then Alarm 22010 :
MD11410 $MN_SUPPRESS_ALARM_MASK, Bit 3 = 0
is output.

Following acknowledgement of gear stage change via the PLC
with NC/PLC IS:
DB31, ... DBX16.3 (gear changed)
the spindle is in speed control mode (DB31, ... DBX84.7).

For further information on the signal exchange between PLC and NC, see Section "A2: Various NC/PLC interface signals and functions (Page 33)".

Second gear stage data set

The automatic gear stage change M40 can be extended by a second configurable gear stage data set.

The second gear stage data set is used **exclusively** in connection with tapping without compensation chuck (G331, G332) so that an effective adjustment of spindle speed and motor torque can be achieved.

The activation is undertaken by setting the following bit for the master spindle:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, Bit 5 = 1

The number of used gear stages of the second gear stage data set is defined with the machine data:

MD35092 $MA_NUM_GEAR_STEPS2

The second gear stage block data set is deactivated if:

MD35092 $MA_NUM_GEAR_STEPS2 = 0 (basic setting)

The first gear stage data set then selects the gear stage when M40 is active.

Note

The number of gear stages in the second data set can vary from the first. If no appropriate gear stage is found for a programmed speed for M40, then no gear stage change is carried out (exceptions, see "M40: Automatic gear stage selection for speeds outside the configured switching thresholds (Page 1489)").

For more information about a typical program sequence in thread cutting without compensating chuck G331/G332 see:
References:
Programming Manual - Fundamentals; Motion Commands

The gear stages 1 to 5 of the second gear stage data set can be parameterized via the following machine data:

Machine data	Meaning
MD35112 $MA_GEAR_STEP_MAX_VELO2[n]	Maximum speed for automatic gear stage change
MD35122 $MA_GEAR_STEP_MIN_VELO2[n]	Minimum speed for automatic gear stage change
MD35212 $MA_GEAR_STEP_POSCTRL_ACCEL2[n]	Acceleration in position control mode

Note

The number of servo parameter sets concerning the mechanical factors remain unchanged. Furthermore, five mechanical gear stages for the spindle and one for the axis operation can be configured.

The speed limitations are configured only once for each gear stage with the following machine data, independently of the different switching thresholds:

Machine data	Meaning
MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[n]	Maximum speed of gear stage
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[n]	Minimum speed of gear stage

For tapping without compensating chuck (G331, G332) the speed can be limited to the linear acceleration range of the motor additionally. For this, the maximum speed of the linear motor characteristics range is specified in the following machine data as a function of the gear stage:

MD35550 $MA_DRILL_VELO_LIMIT[n]

Specify gear stage in the part program

Automatic selection with active M40

The gear stage is automatically selected by the control. The gear stage in which the programmed spindle speed (S...) is possible is checked in this context. If a gear stage results from this that is not equal to the current (actual) gear stage, then the following NC/PLC interface signals are set:

DB31, ... DBX82.3 (change over gear stage)

DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)

While the appropriate gear stage is being determined, a gear stage change is only requested if the new speed is not within the permissible speed range of the active gear stage.

The speed is limited to the maximum speed of the current gear stage or raised to the minimum speed of the current gear stage and the appropriate NC/PLC interface signal is set:

DB31, ... DBX83.1 (speed setpoint limited)

DB31, ... DBX83.2 (speed setpoint increased)

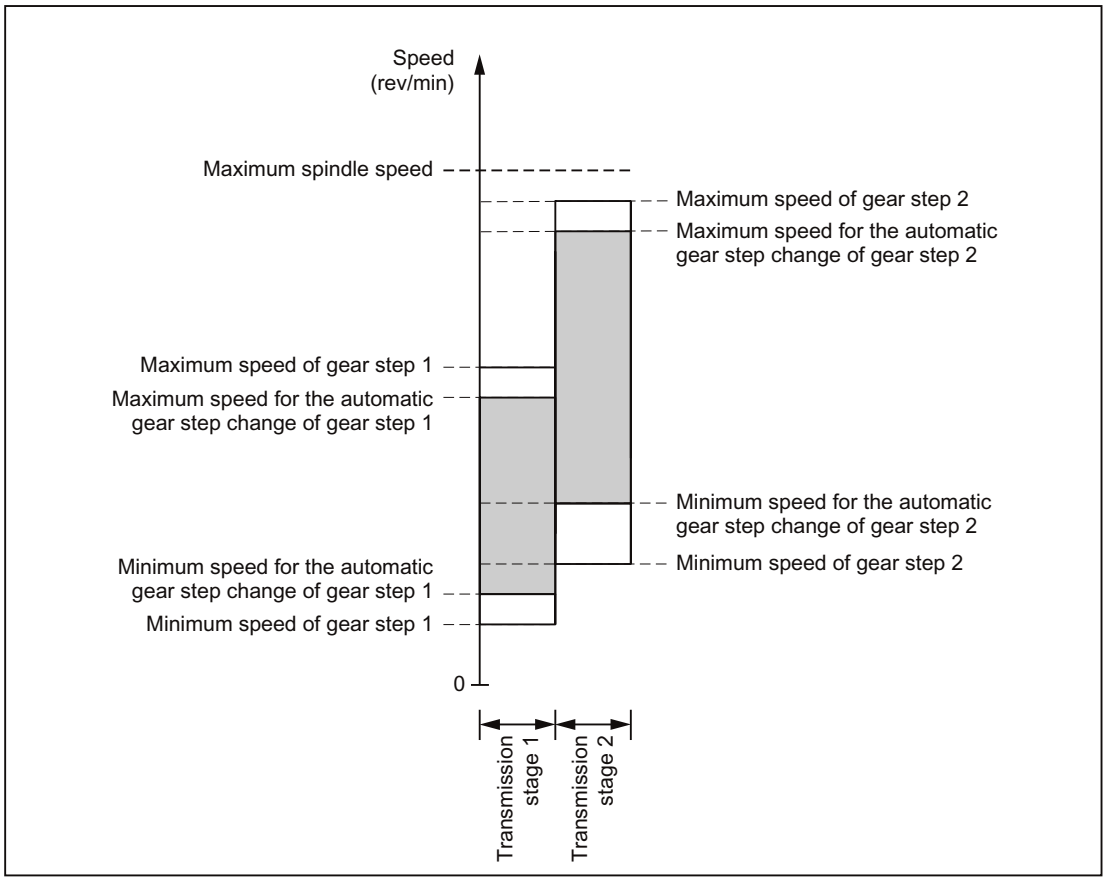


Figure 15-4 Example for two gear stages with overlapping speed ranges for automatic gear stage change (M40)

Note

In the case of M40, the spindle must be in open-loop control mode for automatic gear stage selection with an S word. Otherwise the gear stage change is rejected and the following alarm is set:

Alarm 22000 "gear stage change is not possible"

Note

An active reduction gear is not considered in the selection for the automatic gear stage change.

Permanently defining the gear stage with M41 to M45

The gear stage can be permanently defined in the part program with M41 to M45.

If a gear stage is specified via M41 to M45 that is not equal to the current (actual) gear stage, then the following NC/PLC interface signals are set:

DB31, ... DBX82.3 (change over gear stage)

DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)

The programmed spindle speed (S...) then refers to this permanently defined gear stage:

- If a spindle speed is programmed and it is higher than the maximum speed of the permanently defined gear stage (MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT), then the speed is decreased to this limit and the following NC/PLC interface signal is set:

  DB31, ... DBX83.1 (speed setpoint limited)

- If a spindle speed is programmed and it is lower than the minimum speed of the permanently defined gear stage (MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT), then the speed is increased to this minimum and the following NC/PLC interface signal is set:

  DB31, ... DBX83.2 (speed setpoint increased)

Block change

When programming the gear stage change in the part program, the gear stage change set remains active until it is aborted by PLC.

This corresponds to the effect as if the following NC/PLC interface signal were set:

DB21, ... DBX6.1 (read-in disable)

Specification of gear stage via PLC with FC18

The gear stage change can also be performed by function block FC18 during a part program, in the reset state or in all operating modes.

If the speed and direction of rotation is specified with FC18, the NC can be requested to select the gear stage as appropriate for the speed. This corresponds to an automatic gear stage change with M40.

The gear stage is not changed if:

- The spindle is positioned via FC18.
- The spindle is traversed in the axis mode.

For further information on the FC18 function block, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)".

Specification of a gear stage in synchronized actions

The gear stage change can be requested by synchronized actions using:

- `DO M40 S...`

  Automatic gear stage selection to the programmed speed `S...`

- `DO M41... M45`

  Direct selection of gear stages 1 ... 5

- `DO M70`

  For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5

  (see "Configurable gear step in M70 (Page 1456)")

The gear stage is **not** changed if:

- The spindle is positioned via synchronized actions.
- The spindle is traversed in the axis mode.

Note

For further details, please refer to the section "Specification of a gear stage in part program".

Exception:

The block change is not affected by the specification of a gear stage in synchronized actions.

Manual specification of a gear stage

Outside a part program that is running, the gear stage can also be changed without a request from the NC or the machine. This is the case, for example, when a gear stage is changed manually.

To select the appropriate parameter set, the NC must be informed of the current gear stage. To enable this, the control or the part program must be in the reset state.

Supplementary conditions

Transfer of the gear stage to the NC is initiated when
NC/PLC IS:
DB31, ... DBX16.0-16.2 (actual gear stage A to C) changes.

These three bits must be set continuously during operation.

Successful transfer is acknowledged with NC/PLC IS:
DB31, ... DBX82.0-82.2 (set gear stage A to C)
to the PLC.

NC/PLC IS:
DB31, ... DBX16.3 (gear changed)
must not be set.

If position control is active when a new gear stage is specified by the PLC with DB31, ... DBX16.0-16.2, then it is switched off for the duration of this changeover sequence.

NC stop during gear stage change

The spindle cannot be stopped with NC/PLC IS:
DB21, ... DBX7.4 (NC stop)
if:

- The spindle is not yet in oscillation mode for the gear stage change.
- NC/PLC IS:
  DB31, ... DBX16.3 (gear changed)
  is not set.

Note

Options for aborting:

DB31, ... DBX2.2 (delete distance-to-go / spindle reset)
or
DB31, ... DBX16.3 (gear changed)
with corresponding acknowledgement from actual gear stage:
DB31, ... DBX16.0-16.2 (actual gear stage).

Spindle response after a gear stage change

How the spindle behaves once the gear stage has been changed depends on the following initial conditions:

- If the spindle was in the stop state before the gear stage change (M5, FC18: "Stop rotate spindle"), in positioning or axis mode, M5 (spindle stop) is active after completion of the gear stage change.
- If a direction of rotation
  (M3, M4, FC18: "Start spindle rotation"), then the last speed and direction of rotation will become active again after the gear stage change. In the new gear stage, the spindle accelerates to the last spindle speed programmed (`S...`).
- If position control was active before the gear stage change (`SPCON`), then it is reactivated after the gear stage change.

  The next block in the part program can be executed.

Special features

The following points must be observed on gear stage change:

- The gear stage change is not terminated by selecting NC/PLC IS:
  DB31, ... DBX20.1 (run-up switchover to V/f mode).

  Setpoint 0 is output.

  The gear stage change is acknowledged as usual via the NC/PLC interface signal:

  DB31, ... DBX16.3 (gear is changed)
- The "Ramp-function generator rapid stop" signal must be reset by the PLC before the gear stage change is completed by the PLC.
- The process sequence of the gear stage change is ended during NC reset without any alarm output.

  The gear stage output with NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  is applied by the NC.

Star/delta switchover with FC17

Digital main spindle drives can be switched in both directions between star and delta using FC17, even when the spindle is running. This automatic switchover is controlled by a defined logic circuit in FC17 which provides the user with a configurable switchover time for the relevant spindle.

For further information on the FC17 function block, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 907)".

15.4.2 Spindle gear stage 0

Technical background

For machine's where the spindle load gear can be changed over, situations can occur where the gear train between the motor and load (workpiece/tool) is interrupted. This state can occur, e.g. when pressing Reset or Emergency Stop while performing a gear stage change or when the machine is commissioned for the first time while it is being installed. The control must identify this state where the gear train is open and the next gear stage change request must be unconditionally executed.

Function

When the gear is disengaged, the binary-coded value "0" (≙ gear stage 0) is transferred to the NC from the PLC using the interface signal bits DB31, ... DBX16.0-2 (actual gear stage A to C):

DB31, ... DBX16.0-2 = 0

The value is used by the control to identify the state where the gear train is open.

Effects on the gear stage change

Gear stage change in the part program

The actual gear stage signaled from the PLC is read by the NC when starting a part program. If, at this instant in time, a value of "0" is read for the actual gear stage, then the next gear stage change is executed and the gear stage change dialog is performed by the PLC. If a value greater than "0" is read, then already in the program a comparison is made between the requested and active gear stage. If both gear stages are the same, the gear stage is not changed and a possibly programmed path motion is not interrupted.

Gear stage change in synchronized actions, FC18 and DBB30

The actual gear stage signaled from the PLC is always evaluated by the NC when the gear stage is changed. The gear stage is always changed if a value of "0" is read from the NC. When reading a value greater than "0", a comparison is made between the requested and active gear stage. The gear stage is only changed with the PLC if the two values are not equal and the NC/PLC interface signal DB31, ... DBX82.3 (change over gear) is then output.

Boundary conditions

- **Output of DB31, ... DBX16.0-2 = 0**

  When the gear is disengaged, the PLC must enter gear stage 0 in the NC/PLC interface DB31, ... DBX16.0-2 (actual gear stage A to C).

- **Enabling the gear stage change**

  The precondition for a gear stage change after reaching gear stage 0 is the general enable of the gear stage change via via machine data:

  MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear stage change)

  MD35090 $MA_NUM_GEAR_STEPS (number of gear stages set up)

  MD35092 $MA_NUM_GEAR_STEPS2 (2nd gear stage data set: Number of gear stages that have been created) if MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, bit 5 = 1 (tapping without compensating chuck)

- **PLC user program/ POWER ON ASUB**

  The PLC user program or POWER ON ASUB should ensure that when the gear is disengaged (gear stage 0) before spindle motion, a gear stage change request is programmed. For instance, this can be realized with `M41` in the ASUB. Spindle motion such as e.g. in JOG or in axis operation does not generate any gear stage change itself.

Example

Example for the sequence to select the first gear stage after POWER ON

1. POWER ON.
2. The PLC user program determines, in the mechanical environment, the "Gear is disengaged" state.
3. The PLC transfers the "Gear is disengaged" state to the NC by setting:

   DB31, ... DBX16.0-2 = 0
4. Part program start or POWER ON ASUB.
5. N05 (part program, refer below) is executed:

   The gear stage is changed to gear stage 1.

   From the NC:

   – the following NC/PLC-interface signal is set:

     DB31, ... DBX82.3 (change over gear stage)

   – the setpoint gear stage 1 is signaled to the PLC:

     DB31, ... DBX82.0 = 1

     DB31, ... DBX82.1 = 0

     DB31, ... DBX82.2 = 0

6. Mechanical gear stage change, acknowledgement

   If the gear stage is selected, then from the PLC:

   – the following NC/PLC-interface signal is set:

     DB31, ... DBX16.3 (gear is changed)

   – Actual gear stage 1 signaled to the NC:

     DB31, ... DBX16.0 = 1

     DB31, ... DBX16.1 = 0

     DB31, ... DBX16.2 = 0

7. N80 is executed:

   Due to the optimization of the gear stage change frequency in the part program, the gear stage is not changed.

Part program:

```
Program code            Comment
N05 M41                 ; Select 1st gear stage
...
N80 M41                 ; No gear stage change, if the 1st gear stage is selected.
```

Configuring data for spindle 1 (AX5):

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE[AX5] = 1 (enable gear stage change)

15.4.3 Determining the spindle gear stage

The actual stage of a spindle can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variables:

$VC_SGEAR[<n>]	Currently selected spindle gear stage $VC_SGEAR reads the actual gear stage signaled from the PLC. Range of values: 0 ... 5
$AC_SGEAR[<n>]	Active spindle gear stage $AC_SGEAR reads the setpoint gear stage in the main run. Range of values: 1 ... 5 The data set for the spindle is activated corresponding to this gear stage.

Note

For a search, the actual gear stage ($VC_SGEAR[<n>]) can differ from the setpoint gear stage ($AC_SGEAR[<n>]) as, during the search, no gear stage change takes place. Therefore, using $VC_SGEAR[<n>] and $AC_SGEAR[<n>], it can be interrogated whether a gear stage change should be made after a search.

- Without preprocessing stop in the part program via system variables:

$P_SGEAR[<n>]	Setpoint gear stage $P_SGEAR reads the gear stage programmed in the part program (`M41` ... `M45`), for `M40` selected, or for `M70`, the configured gear stage.
$P_SEARCH_SGEAR[<n>]	Search: Gear-specific M function $P_SEARCH_SGEAR contains the last programmed gear stage M function collected with the block search.

15.4.4 Parameter set selection during gear step change

Servo parameter sets

The servo parameter sets 1 to 6 adapt the position controller to the changed properties of the machine during a gear change of the spindle.

Parameter set selection during gear stage change

The gear stage parameter set (interpolation parameters) and, depending on the setting in the following machine data, the servo parameter set are also modified during gear stage change.

MD35590 $MA_PARAMSET_CHANGE_ENABLE (parameter set change possible)

Value	Meaning
0	In-system parameter set selection The parameter sets of the servo are assigned permanently. The following applies: • For axes and spindles in the axis mode, the first parameter set is active in principle. Exception: For G33, G34, G35, G331 and G332, for the axes involved, the parameter set with the following number is activated: Master spindle gear stage + 1 (corresponds to parameter set No. 2 ... 6) • For spindles in the spindle mode, the parameter set is set matching the gear stage.
1	Besides the in-system parameter set selection, there is also the option of an "external" parameter set selection. • By the PLC (DB31, ... DBX 9.0 - 9.2) • Via programming of SCPARA in the part program or in synchronized actions However, the in-system parameter set selection has priority. Note: Value 1 is relevant only to axes.
2	The servo parameter set is specified **exclusively** by the PLC (DB31, ... DBX 9.0 - 9.2) or through the programming of SCPARA in the part program or in synchronized actions (for axes and spindles). The 1st parameter set is selected after POWER ON.

Spindle mode

MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0 or 1

The parameter set is selected according to the gear stage + 1.

The active gear stage is located in:

DB31, ... DBX16.0-16.2 (actual gear stage A to C)

The active parameter set is output in:

DB31, ... DBX69.0-69.2 (controller parameter set A to C)

One set of parameters, with the following assignment, is provided by the NC for each of the five gear stages:

Data set for ...	NC/PLC interface DBX 69.2 / 69.1 / 69.0	Parameter set Number	Parameter set Index [n]
Axis mode	Last active gear stage	1	0
Gear stage 1	001	2	1
Gear stage 2	010	3	2
Gear stage 3	011	4	3
Gear stage 4	100	5	4
Gear stage 5	101 110 111	6	5

Spindle in axis mode

If the spindle is in axis mode, the parameter set index "0" is selected in the servo (note MD35590 $MA_PARAMSET_CHANGE_ENABLE!).

The gear stage change behavior depends on the setting in the machine data:

MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE (gear stage for axis mode in M70)

If there is no gear stage configured for axis mode (MD35014 = 0), no implicit gear stage change takes place in M70 (default setting!). The last gear stage is saved internally and is reactivated with the associated parameter set during the next spindle programming.

If, however, a gear stage is configured for axis mode (MD35014 = 1 ... 5), a gear stage change to gears 1 ... 5 takes place during the execution of M70. When changing from axis mode to spindle mode, the gear stage loaded with M70 remains activated. The gear stage which is activated in spindle mode prior to M70 is not automatically loaded again.

See also "Configurable gear step in M70 (Page 1456)".

Load gearbox transmission ratio

It is possible to configure positive or negative **load gearbox factors** for each gear stage and in axis mode.

The setting is undertaken separately for numerator and denominator via the machine data:

MD31050 $MA_DRIVE_AX_RATIO_DENOM[n] (load gearbox denominator)

MD31060 $MA_DRIVE_AX_RATIO_DENOM[n] (load gearbox numerator)

The setting range is the same size for positive and negative load gearbox factors.

It is not possible to enter the value "0".

Note

If an indirect encoder is configured, and the load gearbox transmission ratio changes, then the reference is lost and the NC/PLC interface signal:
DB31, ... DBX60.4/60.5 (referenced / synchronized 1 or 2)
is reset for the relevant measuring system.

References

For further information about control and servo parameter set, please refer to:

- Functions Manual - Basic Functions; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)
- Programming Manual, Job Planning; Section: Programmable servo parameter set

15.4.5 Intermediate gear

Application and functions

A configured intermediate gear can be used to adapt a variety of rotating tools. The intermediate gear on the tool side has a multiplicative effect on the motor/load gearbox.

It is set via machine data:

MD31066 $MA_DRIVE_AX_RATIO2_NUMERA (intermediate gear numerator)

MD31064 $MA_DRIVE_AX_RATIO2_DENOM (intermediate gear denominator)

An encoder on the tool for the intermediate gear
is configured with machine data:
MD31044 $MA_ENC_IS_DIRECT2 (Encoder on intermediate gear)
.

Change parameters for these machine data can be activated with "NewConfig" either using the SinuCOM-NC commissioning software or via a softkey on the operator panel (HMI). The existing motor/load gearboxes, on the other hand, are active after POWER ON.

Tool change

If the intermediate gear is changed at the same time as the tool, the user must also reconfigure the transmission ratio of the numerator and denominator via the machine data of the intermediate gear.

Example:

In the case of an installed tool with a transmission ratio of 2:1, a suitable intermediate gear is configured and is activated immediately in the part program with the command NEWCONF.

Program code

```
N05 $MA_DRIVE_AX_RATIO2-NUMERA[AX5] = 2
M10 $MA_DRIVE_AX_RATIO2-DENOM[AX5] = 1
N15 NEWCONF
```

⚠ CAUTION

It remains the task of the user to stop within the appropriate period in order to make changes to the machine data when required and then activate a "NewConfig".

Switchover

Switchover to a new transmission ratio is performed immediately by means of NewConfig. From a technological viewpoint, the associated mechanical switchover process takes some time, since, in mechanical terms, a different intermediate gear with rotating tool is being installed.

Note

At zero speed, switchover is jerk-free. The user is therefore responsible for taking appropriate precautions.

Applications in which switchover takes place during motion and which require smoothed or soft speed transition can be handled using existing setpoint speed filters.

For further explanations regarding control engineering dependencies, see Section "G2: Velocities, setpoint / actual value systems, closed-loop control (Page 331)".

15.4.6 Nonacknowledged gear step change

Mode change

A gear stage change that has not been acknowledged cannot be interrupted by a change in operating mode (e.g. switchover to JOG).

The switchover is delayed by the maximum period entered in machine data:
MD10192 $MN_GEAR_CHANGE_WAIT_TIME
.

If the gear stage change is not acknowledged within this time, the NC will output an alarm:

Further events

Events that initiate reorganization will also wait until a gear stage change is completed.

The time entered in machine data:
MD10192 $MN_GEAR_CHANGE_WAIT_TIME
determines how long the control waits before executing the gear stage change.
If this time elapses without the gear stage change being completed, the NC responds with an alarm.

The following events have an analog response:

- User ASUB
- Mode change
- Delete distance-to-go
- Axis interchange
- Activate PI user data
- Enable PI service machine data
- Switch over skip block, switch over Dry Run
- Editing in the modes
- Compensation block alarms
- Overstore
- Rapid retraction with G33, G34, G35
- Subprogram level abort, subprogram abort

Response after POWER ON

The active gear stage on the machine can be specified by the PLC after POWER ON and in the RESET state.

The NCK will then select the appropriate parameter set
and check back the NC/PLC interface signals:
DB31, ... DBX82.0-82.2 (set gear stage A to C)
to the PLC.

15.4.7 Gear step change with oscillation mode

What is oscillation?

Oscillation in this context means that the spindle motor rotates alternately in the clockwise and counter-clockwise directions. This oscillation movement makes it easy to engage a new gear stage.

Oscillation mode

NC/PLC IS:
DB31, ... DBX82.3 (change gear)
displays that a gear stage change is required.

In principle, the new gear stage can also be engaged without oscillation

1. MD35010 $MA_GEAR_STEP_CHANGE_ENABLE must be set to 1.
2. NC/PLC interface signal DB31, ... DBX84.6 (oscillation mode) is set.
3. The acceleration is set in the machine data:

   MD35410 $MA_SPIND_OSCILL_ACCEL

DB31, ... DBX18.5 (oscillation speed)

The spindle is in oscillation mode if a new gear stage was defined using automatic gear stage selection (M40) or M41 to M45 (DB31, ... DBX82.3 (change gear) is set).

NC/PLC IS:
DB31, ... DBX82.3 (change gear)
is only enabled when a new gear stage is defined
that is not the same as the current actual gear stage.

If NC/PLC IS:
DB31, ... DBX18.5 (oscillation speed)
is simply set by the PLC without a new gear stage being defined by the NC,
the spindle does not change to oscillation mode.

Oscillation mode is activated with NC/PLC IS:
DB31, ... DBX18.5 (oscillation speed)
.

Depending on NC/PLC IS:
DB31, ... DBX18.4 (oscillation via PLC)
while the function is in operation, a distinction is made between:

- Oscillation via NCK
- Oscillation via PLC
- Oscillation with FC 18

  References:
  Function Manual, Basic Functions; PLC Basic Program (P3)

Oscillation time

The oscillation time for oscillation mode can be defined in a machine date for each direction of rotation:

Oscillation time in M3 direction (referred to as t1 in the following):	MD35440 $MA_SPIND_OSCILL_TIME_CW
Oscillation time in M4 direction (referred to as t2 in the following):	MD35450 $MA_SPIND_OSCILL_TIME_CCW

Oscillation via NCK

Phase 1:

NC/PLC IS:
DB31, ... DBX18.5 (oscillation speed)
accelerates the spindle motor to the speed (with oscillation acceleration) defined in machine data:
MD35400 $MA_SPIND_OSCILL_DES_VELO (oscillation speed)
.

Start direction is defined through the following machine data:
MD35430 $MA_SPIND_OSCILL_START_DIR (start direction with oscillation)

The time t1 (or t2) is started
according to which start direction is given in the machine data:
MD35430 $MA_SPIND_OSCILL_START_DIR

The time - and not the fact that the oscillation speed is reached - is always decisive.

Phase 2:

If time t1 (t2) has elapsed, the spindle motor
accelerates in the opposite direction to the speed defined in machine data:
MD35400 $MA_SPIND_OSCILL_DES_VELO
.

Time t2 (t1) starts.

Phase 3:

If time t2 (t1) has passed, the spindle motor accelerates in the opposite direction (same direction as in Phase 1) to the speed defined in machine data:
MD35400 $MA_SPIND_OSCILL_DES_VELO
.

Time t1 (t2) starts. The process continues with Phase 2.

Oscillation via PLC

NC/PLC IS:
DB31, ... DBX18.4 (oscillation via PLC)
and
DB31, ... DBX18.5 (oscillation speed)
accelerates the spindle motor to the speed (with oscillation acceleration) defined in machine data:
MD35400 $MA_SPIND_OSCILL_DES_VELO (oscillation speed)
.

The direction of rotation is defined by NC/PLC IS:
DB31, ... DBX18.7 (set direction of rotation CCW)
and
DB31, ... DBX18.6 (set direction of rotation CW)
.

The oscillation movement and the two times t1 and t2 (for clockwise and counter-clockwise rotation) must be simulated on the PLC.

Special features

Setting/Resetting the NC/PLC IS and machine data in oscillation mode:

- To decelerate the spindle, the PLC user need not set NC/PLC IS:
  DB31, ... DBX4.3 (spindle stop)
  .

  The spindle is brought to a standstill internally by the control when a gear stage change is requested.
- The gear stage change should always be terminated with NC/PLC IS:
  DB31, ... DBX16.3 (gear changed)
  .
- NC/PLC IS:
  DB31, ... DBX18.5 (oscillation speed)
  should be used to support mechanical engagement of the gear.

  It has no effect on the internal control mechanism for the gear stage change procedure and should therefore only be set as necessary.

- If NC/PLC IS:
  DB31, ... DBX18.5 (oscillation speed)
  is reset, oscillation mode stops.

  However, the spindle remains in "oscillation mode".
- The acceleration is defined in the following machine data:
  MD35410 $MA_SPIND_OSCILL_ACCEL
- The spindle will cease to be synchronized if an indirect measuring system (motor encoder) is used.

  If the machine data is set to:
  MD31050 $MA_ENC_IS_DIRECT = 0,
  NC/PLC IS:
  DB31, ... DBX60.4/5 = 0 (referenced/synchronized)
  is automatically deleted.

  The zero mark is synchronized the next time it is crossed.

End of oscillation mode

On termination of oscillation mode, the spindle returns to open-loop control mode and automatically changes to the mode defined by `SPCON` or `SPCOF`.

All gear-specific limit values (min./max. speed, etc.) correspond to the set values of the actual gear stage.

Functionality

Machine tools of conventional design require a gear stage of the spindle in oscillation mode.

If the machine data configuration is:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 1
the following sequence is implemented:

- Deceleration of the spindle.

  The braking action corresponds to an M5 movement.
- Output of VDI interface signals:
  DB31, ... DBX84.6 (oscillation mode)
  DB31, ... DBX82.3 (change gear)
  DB31, ... DBX82.0-82.2 (set gear stage A to C).

  If position control has been enabled, it is disabled:
  DB31, ... DBX61.5 = 0.
- The load gearbox can now "disengage".

- NC/PLC IS:
  DB31, ... DBX18.5 (oscillation enable)
  can be set by the PLC.

  The spindle motor then performs an oscillation motion with preset values.
  The oscillation motion is designed to facilitate and accelerate the re-engaging of the gear wheels.
- Writing of NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  by the PLC.
- Once the PLC has sent:
  DB31, ... DBX16.3 (gear changed)
  to the NCK, the last movement to be active is continued, if available.

  For indirect encoders (motor encoders), the homing status is cleared:
  DB31, ... DBX60.4/5 = 0.

Block change

If the spindle is switched to oscillation mode
with NC/PLC IS:
DB31, ... DBX82.3 (change gear),
the processing of the part program remains suspended.
A new block is not executed.

If oscillation mode is terminated with NC/PLC IS:
DB31, ... DBX16.3 (gear changed),
the processing of the part program is resumed.
A new block is executed.

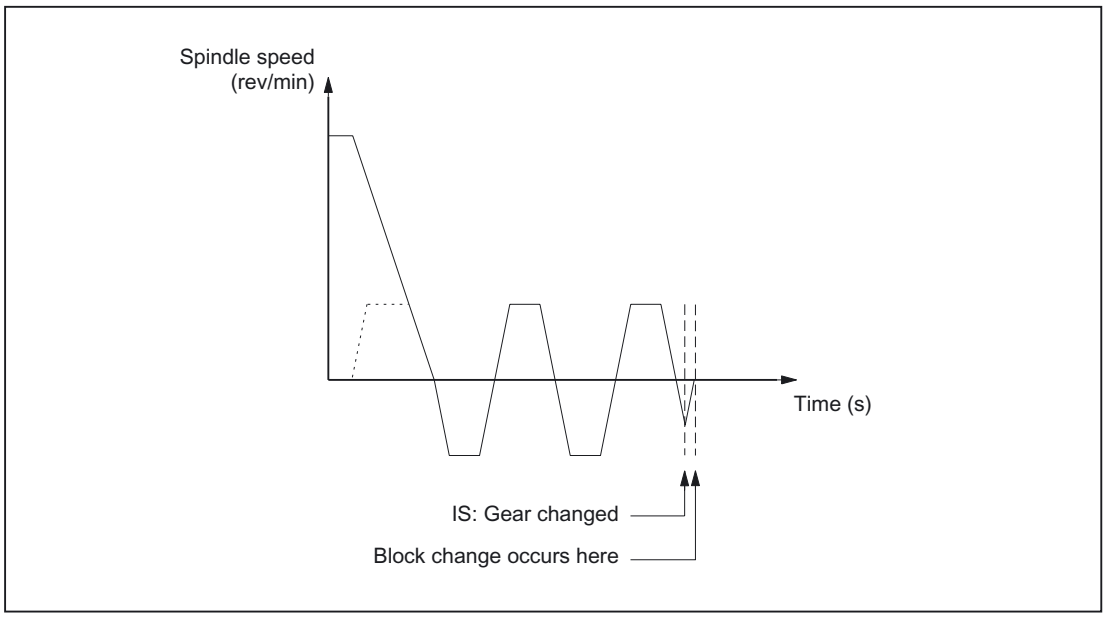
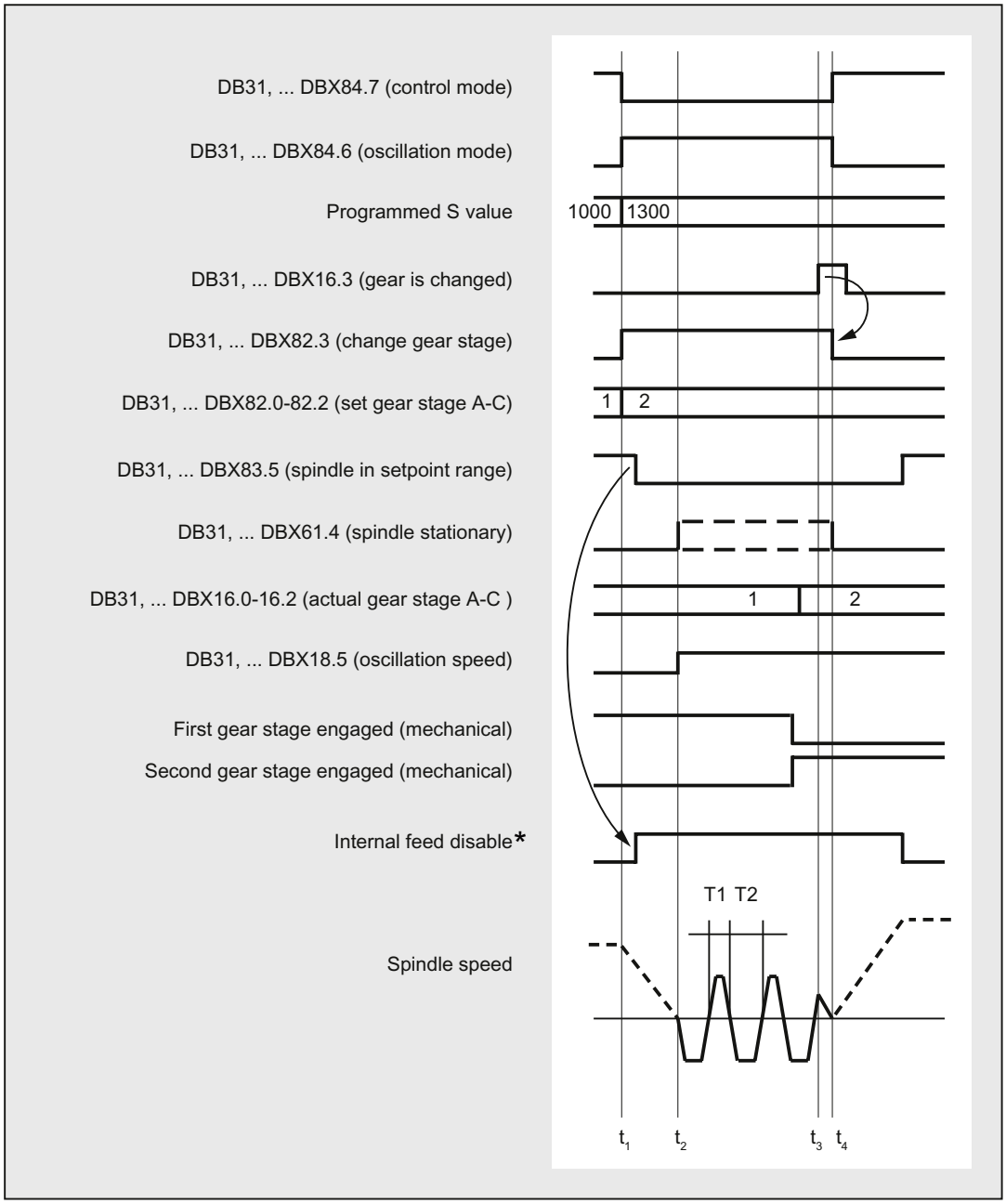


Figure 15-5 Block change following oscillation mode

Oscillation mode

Typical time sequence for the gear stage change with a spindle:



t_1: With the programming of S1300, NCK detects a new gear stage (second gear stage), sets NST DB31, ... DBX82.3 (change gear) and blocks processing for the next part program block (= internal feed disable*).

t_2: The spindle is stationary, and oscillation starts (oscillation via the NCK). NST DB31, ... DBX18.5 (oscillation speed) must be set at the latest by t_2.

t_3: The new gear stage is engaged. The PLC user transfers the new (actual) gear stage to the NCK and sets NST DB31, ... DBX16.3 (gear is changed).

t_4: NCK then retracts NST DB31, ... DBX82.3 (change gear), ends the oscillation, releases the next part program block for processing, and accelerates the spindle to the new S value (S1300).

* : The internal feed disable is set if:

- The spindle gear stage change has been programmed via the part program **and**
- A processing block is activated (i.e. G0 is not active)

The internal feed disable is **not** set during a gear stage change from synchronized actions or in the case of specifications via the PLC with FC 18.

Figure 15-6 Gear stage change with stationary spindle

15.4.8 Gear stage change at fixed position

Application and advantages

Machine tools increasingly use standardized spindle drives, firstly to save technological dead time on a gear stage change and secondly to gain the cost benefits of using standardized components.

The "Gear stage change at fixed position" function supports the "directed gear stage change" of load gearboxes that need to be activated in a different way than the NC. The gear stage change can in this case only be performed at a defined spindle position. An oscillation movement as required by conventional load gearboxes is thus no longer necessary.

Sequence for gear stage change at fixed position

The gear stage change at fixed position

Machine data configuration:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2
runs the following sequence:

- Positioning of the spindle from standstill or movement to the position configured in machine data:
  MD35012 $MA_GEAR_STEP_CHANGE_POSITION.

  If the gear stage change is performed out of a movement, then the current direction of rotation is maintained. The spindle is in positioning mode during the positioning action.

  NC/PLC IS:
  DB31, ... DBX84.5 (positioning mode)
  is output.

  If no reference is available:
  DB31, ... DBX60.4/5 = 0
  or NC/PLC IS:
  DB31, ... DBX17.4/5 (resynchronize on positioning MS 1/2)
  is set, the positioning action is extended by the time it takes to find the zero mark.
- After reaching the gear stage change position configured in machine data:
  MD35012$MA_GEAR_STEP_CHANGE_POSITION
  the machine waits for the time in machine data:
  MD35310 $MA_SPIND_POSIT_DELAY_TIME
  before switching to oscillation mode,
  and the known gear stage change dialog starts.
- Output of NC/PLC interface signals:
  DB31, ... DBX84.6 (oscillation mode)
  DB31, ... DBX82.3 (change gear)
  DB31, ... DBX82.0-82.2 (set gear stage A to C).
- Position control is not disabled when an active measuring system with indirect encoder (motor encoder) is used:

  MD31040 $MA_ENC_IS_DIRECT = 0

  If a measuring system with a direct encoder (load encoder) is active, position control is deactivated:
  DB31, ... DBX61.5 = 0,
  because the induction flux to the load is interrupted and closed-loop position control is no longer possible.
- If position-controlled operation is not possible, it can be disabled by resetting "Controller enable":
  DB31, ... DBX2.1 = 0
  .

- Mechanical switchover of the gear stage on the machine.

  No oscillation movement is required from the drive.

  NC/PLC IS:
  DB31, ... DBX18.5 (oscillation enable)
  and
  DB31, ... DBX18.4 (oscillation via PLC)
  should **not** be set.

  In principle, oscillation movement is still possible at this point.

- Writing of NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  by the PLC.

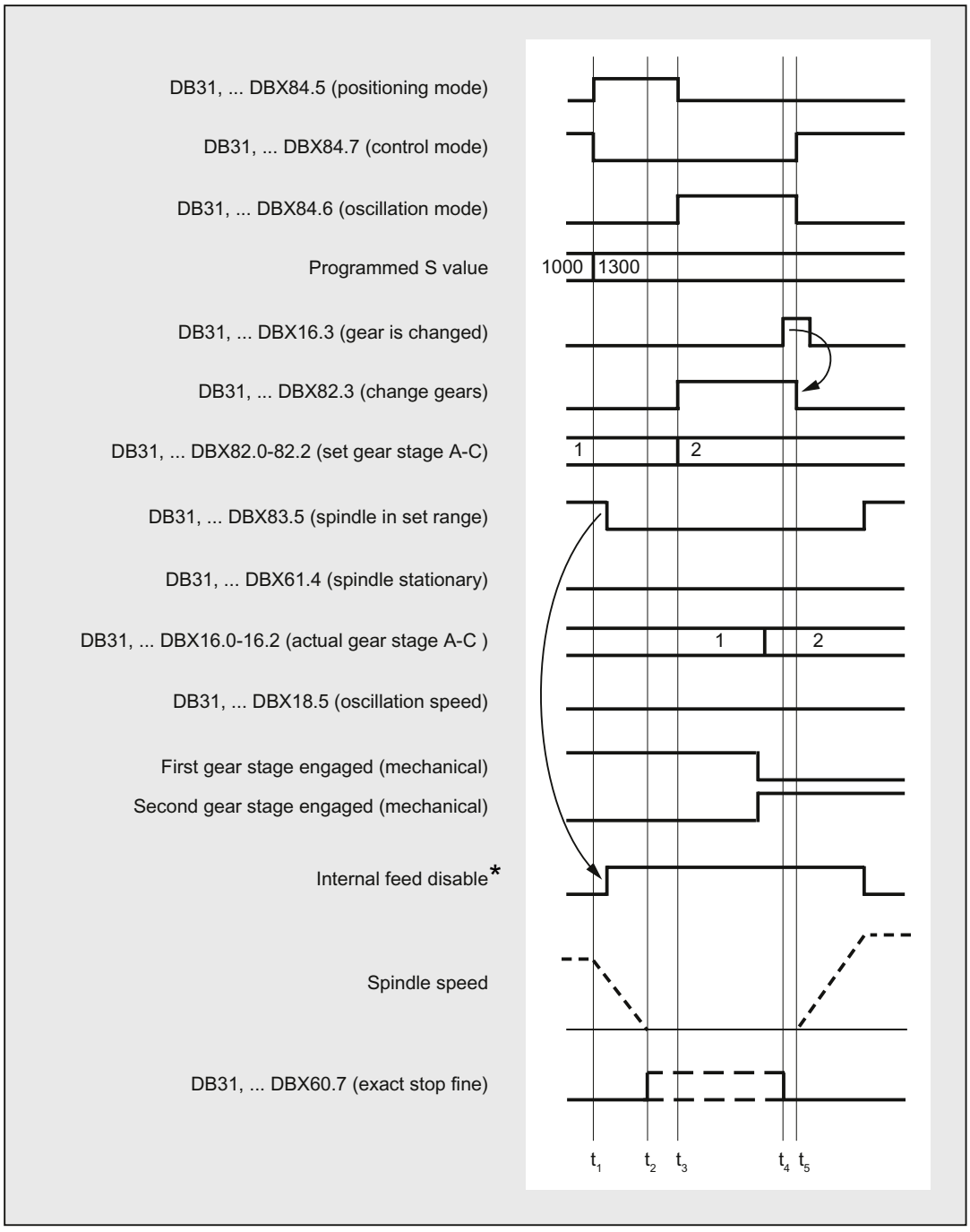
- After signal:
  DB31, ... DBX16.3 (gear stage changed),
  the last movement to be active is continued, if available.

  For indirect encoders (motor encoders), the referencing status is cleared:
  DB31, ... DBX60.4/5 = 0.

  The spindle is in speed control mode and NC/PLC IS:
  DB31, ... DBX84.7 (open-loop control mode)
  is output.

GSC at fixed position

Typical time sequence for the gear stage change at fixed position:



t_1: With the programming of S1300, NCK detects a new gear stage (second gear stage), NCK sets NST DB31, ... DBX84.5 (positioning mode) and blocks processing for the next part program block (= internal feed disable*).

t_2: The spindle is stationary, and exact stop is signaled.

t_3: Gear stage change - wait time

t_4: The new gear stage is engaged. The PLC user transfers the new (actual) gear stage to the NCK and sets NST DB31, ... DBX16.3 (gear is changed).

t_5: NCK then retracts NST DB31, ... DBX82.3 (change gear), releases the next part program block for processing, and accelerates the spindle to the new S value (S1300).

* : The internal feed disable is set if:

- The spindle gear stage change has been programmed via the part program **and**
- A processing block is activated (i.e. G0 is not active)

The internal feed disable is **not** set during a gear stage change from synchronized actions or in the case of specifications via the PLC with FC 18.

Figure 15-7 Gear stage change with stationary spindle

gear stage change position MD35012

The gear stage change position is defined in machine data:
MD35012 $MA_GEAR_STEP_CHANGE_POSITION
for each gear stage.

Gear stage change wait time MD35310

After the positioning action the machine waits for the time configured in machine data:
MD35310 $MA_SPIND_POSIT_DELAY_TIME
until gear change request:
DB31, ... DBX84.6 (oscillation mode)
DB31, ... DBX82.3 (change gear)
and
DB31, ... DBX82.0-82.2 (set gear stage A to C)
are output.

Position identifiers / position

The position is always approached via the shortest path (corresponds to DC).

If no reference is available and the spindle is in standstill
(e.g. after Power On), then the direction of travel is determined by the following machine data:
MD35350 $MA_SPIND_POSITIONING_DIR

If an adjustable gear stage change position is required, then this can be achieved by writing the machine data and by a subsequent NewConfig.
The change of the MD value can be achieved by the part program or HMI.

If the system is unable to reach the preset position, then alarm 22020 is signaled and the gear stage change dialog between NCK and PLC does not take place in order not to destroy the gears. As this alarm is serious, the part program cannot continue and the cause must be eliminated under all circumstances. Experience has shown that the abortion of positioning is usually due to incorrect MD settings or incompatible PLC signals.

Speed

The positioning speed is taken from the machine data which is configured depending on the gear stage:

MD35300 $MA_SPIND_POSCTRL_VELO

NC/PLC IS "Spindle speed override"/"Feedrate override" at
DB31, ... DBX17.0=0: DB31, ... DBB19)
as well as:
DB31, ... DBX17.0=1: DB31, ... DBB0
are effective as normal for positioning.
The positioning speed can be changed proportionally through the program statement `OVRA[Sn]`.

Note

`OVRA[Sn]` is valid modally. After the gear stage change, a value appropriate for the machining should be reset.

The part program statement `FA[Sn]` does not change the positioning speed during gear stage change.

Acceleration

The acceleration values are determined by the machine data which is configured depending on the gear stage:

MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL

and

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

The acceleration can be changed proportionally by programming `ACC[Sn]`.

Note

`ACC[Sn]` is valid modally. After the gear stage change, a value appropriate for the machining should be reset.

Speed-dependent acceleration

The "knee-shaped acceleration characteristic" is effective as in positioning with `SPOS` or FC18.

Jerk

It is currently not possible to limit the change in acceleration.

End of positioning

The transition between the end of the positioning action (DB31, ... DBX84.5) and the start of oscillation mode (DB31, ... DBX84.6) is defined on reaching "Exact stop fine" (DB31, ... DB60.7) and the time value entered in machine data: MD3510 $MA_SPIND_POSIT_DELAY_TIME
.

The determination of the transition condition has an effect firstly on the gear stage change time and secondly on the accuracy of the approach to the preset gear stage change position.

Block change

The block change is stopped and the machining blocks are not started until the gear stage has been changed by the PLC (DB31, ... DBX16.3).

End of gear stage change

Once the gear stage change has been completed, the spindle returns to open-loop control mode and will automatically change to the closed-loop control mode defined by `SPCON` or `SPCOF`.

All gear-specific limit values (min./max. speed of gear stage, etc.) correspond to the check-back values of the actual gear stage.

Supplementary conditions

- The spindle must have at least one measuring system.
- Position-controlled operation must be possible and must have been activated.
- Generally, it must be possible to execute SPOS from the part program, from a synchronized action or via FC18: “Start spindle positioning” without errors.

Unless all requirements can be met, the function described cannot be used successfully.

Activation

The function of gear stage change at fixed position
is activated by the configuration:
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2

15.4.9 Configurable gear step in M70

Technical background

In some machines the spindle needs to be in a particular gear step during axis mode.

Possible reasons:

- Only one optimization (K_v, feedforward control, filter) to suit a gear step can be found in the servo parameter set for axis mode (index 0). The machine data for this parameter set should not be rewritten.
- There is only one mechanical transmission ratio which, unlike the others, possesses little or no backlash compensation. The spindle can only follow a path motion or transformations (e.g. TRANSMIT) together with other axes in this gear step.

Function

If the function is activated, a predefined gear step is loaded automatically during transition to axis mode.

The gear step change is integrated into the M70 process and occurs after spindle deceleration and before the loading of the servo parameter set with index 0 (note MD35590 $MA_PARAMSET_CHANGE_ENABLE!).

The typical dialog between NC and PLC which occurs during gear step changes is executed in a similar way to programmed gear step changes (M41 ... M45).

Requirements

Gear step changes during transition to axis mode require general enabling of the gear step change via the machine data:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear step change).

MD35090 $MA_NUM_GEAR_STEPS (number of gear steps set up)

Activation/Deactivation

The function is activated / deactivated via the machine data:

MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE (gear step for axis mode in M70)

Value	Meaning
0	No implicit gear step change occurs in M70. The current gear step is retained (default setting!).
1 ... 5	A gear step change to gears 1 ... 5 occurs during the processing of M70.

Boundary conditions

Gear step change at fixed position (MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2)

The "gear step change at fixed position" function is supported. The sequence in M70 is then extended by the time it takes to position the spindle. The position is approached at the current gear step.

Transition to axis mode without programming M70

The control system detects the transition to axis mode automatically from the program sequence (see "Implicit transition to axis mode (Page 1411)") and generates the requisite M70 sequence, including the gear step change, within the control system.

Transition to axis mode with FC 18

Implicit gear step change is not supported in transition to axis mode with the FC 18 ("Start axis"). This requires the right gear step to be engaged by the PLC application before switching to axis mode. The gear step change is also possible with the FC 18 ("Start gear step change").

Change from axis mode to spindle mode

When changing from axis mode to spindle mode, the gear step loaded with M70 remains activated. The gear stage which is activated in spindle mode prior to M70 is not automatically loaded again. The servo parameter set is changed from parameter set 1 (index 0) to parameter sets 2 ... 6 (index 1 ... 5) to suit the gear step (with MD35590 $MA_PARAMSET_CHANGE_ENABLE < 2).

Example

Gear step 4 should be loaded in the case of spindle transition to axis mode.

Configuration: MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE[<spindle identifier>] = 4

```
Program code                Comment
N05 M3 S1000
N10 G1 X100 F1000
N15 M70                     ; Gear step 4 is loaded.
N20 POS[C]=77
N25 ...
```

Note

MD35014 can be changed in the NewConfig. Thus, the gear step being loaded can still be changed in the part program before transition to axis mode, if necessary.

15.4.10 Suppression of the gear stage change for DryRun, program test and SERUPRO

Function

For test feed rate (DryRun), program test and SERUPRO, normally, a gear stage change is not required. This is the reason that it can be suppressed for these functions. The corresponding configuration is realized with bits 0 ... 2 in machine data:

MD35035 $MA_SPIND_FUNCTION_MASK

Dry run feedrate (DryRun)	
Bit 0 = 0	When the DryRun function is active - for part program blocks - gear stages are changed with M40, M41 to M45, or programming via FC18 and synchronized actions.
Bit 0 = 1	When the DryRun function is active - for part program blocks - a gear stage change is suppressed with M40, M41 to M45, programming via FC18 and synchronized actions.
Program test and SERUPRO	
Bit 1 = 0	For active program test / SERUPRO function - for part program blocks - gear stages are selected with M40, M41 to M45, programming via FC18 and synchronized actions.
Bit 1 = 1	For active program test / SERUPRO function - for part program blocks - a gear stage change is suppressed with M40, M41 to M45, programming via FC18 and synchronized actions.

DryRun, program testing and SERUPRO	
Bit 2 = 0	Gear stage change for programmed gear stage is **not** performed subsequently on REPOS after deselection of functions DryRun, Program Test and SERUPRO.
Bit 2 = 1	Gear stage change for programmed gear stage is performed after deselection of functions DryRun and SERUPRO if possible.

Sequence

If a gear stage change is suppressed, if necessary, the interpolator will limit the programmed spindle speed to the permissible speed range of the active gear stage.

NC/PLC interface signals DB31, ... DBX83.2 (setpoint speed increased) and DB31, ... DBX83.1 (setpoint speed limited) generated as a result of this limit are suppressed.

Monitoring by the PLC program is not necessary during DryRun and in dry run feedrate.

When the gear stage change is suppressed, no new gear stage setpoint (DB31,... DBX82.0-82.2) is output to the PLC.

The gear stage change request DB31, ... DBX82.3 (change gear) is also suppressed.

This ensures that no gear stage change information has to be processed by the PLC program.

Determining the last active gear stage

System variable $P_GEAR returns the gear stage programmed in the part program (which may not have been output to the PLC).

System variable $AC_SGEAR can be used to read the last active gear stage from the part program, synchronized actions and at the user interface.

Behavior after deselection

The DryRun function can be deselected within a running part program. Once it has been deselected, the correct gear stage requested by the part program must be identified and selected.

It cannot be assured that the remainder of the part program will run without errors until the correct gear stage has been activated. Any necessary gear stage change is performed in the system REPOS started on deselection if the spindle is in speed control mode. A complete gear stage change dialog takes place with the PLC and the last programmed gear stage is activated.

If, for REPOS, there is a mismatch between the gear stage programmed in the part program and the actual gear stage supplied via the NC/PLC interface, then no gear stage change takes place.

The same applies to the SERUPRO function.

Further explanations regarding the block search function SERUPRO, see:
References:
Function Manual, Basic Functions; Mode Group, Channel, Program Mode, Reset Response (K1)

Boundary conditions

If the gear stage change is suppressed, the output spindle speed moves within the speed range specified by the current gear stage.

The following restrictions apply to the subsequent activation of a gear stage change with REPOS:

- The gear stage change is not activated subsequently if the spindle in the deselection or target block is a command spindle (synchronized action) or PLC spindle (FC18).
- If the gear stage cannot be activated because the spindle is in position or axis mode or a link is active, alarm 22011"Channel%1 block%3 spindle2% Change to programmed gear stage not possible" is signaled.

Example

Gear stage change in DryRun

```
; 1. Activate 1st gear stage for output state
N00 M3 S1000 M41                    ; 1st gear stage is selected
M0                                  ; Part program stops

; PI service: Activate dry run feedrate (DryRun)
                                    ; (Configuring)
N10 M42                             ;  2nd gear stage requested, no gear stage
                                      change takes place
N11 G0 X0 Y0 Z0                     ; Positioning axes
N12 M0                              ; Part program stops

; PI service: Deactivate dry run feedrate (DryRun)
                                    ; REORG and REPOS are performed
                                    ; now the gear stage change to the 2nd gear
                                      stage takes place
N20 G1 Z100 F1000
...
N99 M30                             ; Part program end
```

15.5 Additional adaptations to the spindle functionality that can be configured

Spindle-specific functions are set using machine data:

MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions)

MD35035 is bit coded:

Bit	Meaning	
0 ... 2	Gear stage change behavior for test feedrate (DryRun), program test and SERUPRO See "Suppression of the gear stage change for DryRun, program test and SERUPRO (Page 1458)".	
4	The programmed speed `S...` including speed setpoints FC18 and synchronized actions can be accepted in setting data SD43200 $SA_SPIND_S (speed for spindle start via PLC interface). See "Special spindle motion via the PLC interface (Page 1472)".	
5	For bit 5 = 1, the content of setting data SD43200 $SA_SPIND_S acts as the speed setpoint for JOG. You can use the JOG keys to operate the spindle at the speed defined in SD43200. If the content is zero, other JOG speed setpoints are active(see SD41200 $SN_JOGSPIND_SET_VELO).	
8	The programmed cutting velocity `S...` including setpoints via FC18 and synchronized actions can be accepted in setting data SD43202 $SA_SPIND_CONSTCUT_S (cutting rate for spindle start via the PLC interface). See "Special spindle motion via the PLC interface (Page 1472)".	
10	For the master spindle, the value of the 15th G group (feedrate type) can be accepted in setting data SD43206 $SA_SPIND_SPEED_TYPE (spindle speed type for spindle start via the PLC interface). See "Special spindle motion via the PLC interface (Page 1472)".	
12	Bit 12 = 0	Spindle override is **not** effective for the zero mark search with M19, SPOS or SPOSA = 0.
	Bit 12 = 1	Spindle override is effective for the zero mark search with M19, SPOS or SPOSA = 0.

Bit	Meaning	
19	When programming SPOS and SPOSA, auxiliary function M19 can be implicitly generated and output to the PLC. See "Positioning mode (Page 1390)".	
20	In the case of M70 functionality which is generated in the control, auxiliary function M70 can be implicitly generated and output to the PLC. See "Implicit transition to axis mode (Page 1411)".	
22	Bit 22 = 0	The NC/PLC interface signal DB31, ... DBX17.6 (invert M3/M4) also affects the function "interpolatory tapping (G331/G332)". **Note:** The following must be observed for this setting: • For part programs with G331 and G332, it is necessary to set NC/PLC interface signal DB31, ... DBX17.6 (invert M3/M4) **before part program start** to a stable value. • Existing application-based solutions, e.g. as are used in tapping cycles, must, if required, be adapted. It is known, for example, in tapping cycles, that the spindle direction of rotation can be inverted using MIRROR depending on a cyclic-specific setting data.
	Bit 22 = 1	The NC/PLC interface signal DB31, ... DBX17.6 (invert M3/M4) does **not** affect the function "interpolatory tapping (G331/G332)".

Changes to MD35035 become effective after an NC reset.

15.6 Selectable spindles

Function

The "selectable spindles" function allows you to write part programs with reference to the spindles used ("channel spindle, logical spindle") regardless of the actual assignment of configured spindles ("physical spindles") to a channel.

The physical spindles loaded or unloaded by "axis replacement" no longer have to be specified explicitly in the part program.

Spindle number converter (SD42800)

Each spindle is uniquely mapped to a machine axis using a configurable number:

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX...] = <spindle number>

This number always applies to one spindle, whereby it is of no importance in which channel the spindle is actively handled.

The channel spindles can be switched over because an intermediate level is introduced between the logical spindle numbers used in the part program and the physical spindles existing in the channel.

Every logical spindle used in the part program is assigned a physical spindle in a table comprising setting data:

SD42800 $SC_SPIND_ASSIGN_TAB[<n>] (spindle number converter) = ...

Index <n> corresponds to the programmed spindle number or the programmed address extension. The contents of the particular SD is the physical spindle that is actually available.

The spindle number converter is effective in spindle programming by means of:

- The part program
- Synchronized actions

The spindle number converter has no effect with PLC commands, which use function block FC18. The physical spindle must always be addressed there within the context of the axis.

Logical spindles can be changed over by changing SD42800. The changeover can be made from the part program, from the PLC and/or HMI.

Note

The logical master spindle is contained in setting data SD42800 $SC_SPIND_ASSIGN_TAB[0]. It is only used for display purposes.

This setting data is defined in the part program by SETMS (logical spindle).

Unused spindles are assigned the value 0 in SD42800.

System variables, which are involved in the spindle changeover:

$P_S, $P_SDIR, $P_SMODE, $P_GWPS, $AC_SDIR, §AC_SMODE, $AC_MSNUM, $AA_S

References:
Programming Manual, Job Planning

The converted, physical spindle number is always output as the address extension in the auxiliary function output.

Supplementary conditions

- Switchable channel spindles are **not** a substitute for the Axis replacement function.
- You can only switch spindles, which have been assigned to the channel by means of configuration.
- If spindles, which are presently active in another channel, are designated for switchover, either the "Auto-Get" function is triggered for the physical spindle or alarm 16105 "Assigned spindles do not exist" is output, depending on the configuration variant.
- If SD42800 $SC_SPIND_ASSIGN_TAB[<n>] is specified by the PLC or from HMI, then the channel whose table is changed should be in Reset status or the spindle to be changed should not be used in the running part program.

  A synchronized response can be achieved by means of a STOPRE preprocessor stop.
- The multiple mapping of logical to physical spindles is not prevented in the NC. However, with the display of logical spindle in the user interface, there are ambiguities corresponding to the change table.
- Spindle conversion operates on spindles via FC 18.

Activation

SD42800 $SC_SPIND_ASSIGN_TAB[<n>] is enabled by setting the following machine data setting:

MD20092 $MC_SPIND_ASSIGN_TAB_ENABLE=1

Basic setting SD42800

After switching on the NC with the commissioning switch in setting 1 (delete SRAM) SD42800 $SC_SPIND_ASSIGN_TAB[<n>] is in the basic setting.

The numbers of the logical and physical spindles are identical.

SD42800 $SC_SPIND_ASSIGN_TAB[1] = 1

SD42800 $SC_SPIND_ASSIGN_TAB[2] = 2

SD42800 $SC_SPIND_ASSIGN_TAB[3] = 3

SD42800 $SC_SPIND_ASSIGN_TAB[4] = 4

SD42800 $SC_SPIND_ASSIGN_TAB[5] = 5

...

Example

Spindle configurations:

- Assignment, spindle number and machine axis:

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX4] = 1
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX5] = 2
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX6] = 3
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX7] = 5

- Accepting a machine axis in a channel:

MD20070 $MC_AXCONF_MACHAX_USED[0] = 4
MD20070 $MC_AXCONF_MACHAX_USED[1] = 5
MD20070 $MC_AXCONF_MACHAX_USED[2] = 6
MD20070 $MC_AXCONF_MACHAX_USED[3] = 7

- Specifying the master spindle:

MD20090 $MC_SPIND_DEF_MASTER_SPIND = 1

Spindle number converter:

```
MD20092 $MC_SPIND_ASSIGN_TAB_ENABLE=1          ; Activate spindle number converter

SD42800 $SC_SPIND_ASSIGN_TAB[0]=1              ; Master spindle as configured

SD42800 $SC_SPIND_ASSIGN_TAB[1]=1              ; Basic setting of the table

SD42800 $SC_SPIND_ASSIGN_TAB[2]=2
SD42800 $SC_SPIND_ASSIGN_TAB[3]=3
SD42800 $SC_SPIND_ASSIGN_TAB[4]=0              ; Logical spindle not assigned

M3 S1000           ; Address extension=1, M1=3 S1=1000 is output
                     The spindle with configured No. "1" (No. of the physical master
                     spindle) rotates.
...
...
SD42800 $SC_SPIND_ASSIGN_TAB[1]=5          ; Assignment of logical spindle 1 to
                                             physical spindle 5
SD42800 $SC_SPIND_ASSIGN_TAB[2]=3          ; Assignment of logical spindle 2 to
                                             physical spindle 3.
                                             Notice:physical spindle 3 has now been
                                             assigned twice. When programming
                                             logical spindles 2 and 3, physical
                                             spindle 3 is always addressed. In the
                                             basic machine displays, both spindles
                                             rotate.
SETMS(2)           ; SD42800 $SC_SPIND_ASSIGN_TAB[0] = 2 defined internally by NCK.

...
M5                 ; Master spindle = address extension=2, the unset spindle number
                     is output. M3=5
                     The physical spindle configured with number "3" stops.
...
GET(S4)            ; Alarm 16105, as logical spindle "4" cannot be switched.

...
RELEASE(S1)        ; Channel spindle "1" = physical. Spindle "5" is enabled.

...
M30
```

15.7 Programming

15.7.1 Programming from the part program

Programming statements

Statement	Description	
SETMS:	Master spindle is the spindle specified in the following machine data: MD20090 $MC_SPIND_DEF_MASTER_SPIND (position of deletion of the master spindle in the channel)	
SETMS(<n>):	The spindle with the number <n> is the master spindle (may differ from the initial setting: MD20090 $MC_SPIND_DEF_MASTER_SPIND). The master spindle must be defined for the following functions:	
	• G95:	Revolutional feedrate
	• G96 S.../G961 S...:	Constant cutting rate in m/min or feet/min
	• G97/G971:	CancelG96/G961 and freeze last spindle speed
	• G63:	Tapping with compensating chuck
	• G33/G34/G35:	Thread cutting
	• G331/G332:	Tapping without compensating chuck
	• G4 S...:	Dwell time in spindle revolutions
	• Programming M3, M4, M5, S, SVC, SPOS, M19, SPOSA, M40, M41 to M45, and WAITS without entering the spindle number	
	The current master spindle setting can be retained via RESET/part program end and part program start. The setting is done via the machine data: • MD20110 $MC_RESET_MODE_MASK • MD20112 $MC_START_MODE_MASK	
M3: M<n>=3:	Clockwise spindle rotation for the master spindle Clockwise spindle rotation for spindle number <n>	
M4: M<n>=4:	Counter-clockwise spindle rotation for the master spindle Counter-clockwise spindle rotation for spindle number <n>	
M5: M<n>=5:	Spindle stop without orientation for the master spindle Spindle stop without orientation for spindle number <n>	
S...: S<n>=...:	Spindle speed in rpm for the master spindle Spindle speed in rpm for spindle number <n>	
SVC=...: SVC[<n>]=...:	Cutting rate in m/min or feet/min for the master spindle Cutting rate in m/min or feet/min for the spindle <n>	

Statement	Description
SPOS=...: SPOS[<n>]=...:	Spindle positioning for the master spindle Spindle positioning for spindle number <n> The block change is only performed when the spindle is in position.
SPOSA=...: SPOSA[<n>]=...:	Spindle positioning for the master spindle Spindle positioning for spindle number <n> The block change is executed immediately. Spindle positioning continues, regardless of further part program processing, until the spindle has reached its position.
SPOS=DC(...): SPOS[<n>]=DC(...): SPOSA=DC(...): SPOSA[<n>]=DC(...):	The direction of motion is retained for positioning while in motion and the position approached. When positioning from standstill, the position is approached via the shortest path.
SPOS=ACN(...): SPOS[<n>]=ACN(...): SPOSA=ACN(...): SPOSA[<n>]=ACN(...):	The position is always approached with negative direction of motion. If necessary, the direction of motion is inverted prior to positioning.
SPOS=ACP(...): SPOS[<n>]=ACP(...): SPOSA=ACP(...): SPOSA[<n>]=ACP(...):	The position is always approached with positive direction of motion. If necessary, the direction of motion is inverted prior to positioning.
SPOS=IC(...): SPOS[<n>]=IC(...): SPOSA=IC(...): SPOSA[<n>]=IC(...):	The travel path is specified. The direction of travel is determined from the sign in front of the travel path. If the spindle is in motion, the direction of travel is inverted as necessary to allow traversing in the programmed direction. If the zero mark is crossed during traversing, the spindle is automatically synchronized with the zero mark if no reference is available or if a new one has been requested via an interface signal.
M19: M[<n>]=19:	Positioning the master spindle to the position in SD43240 Positioning spindle number <n> to the position in SD43240 The block change is only performed when the spindle is in position.
M70: M<n>=70:	Bring spindle to standstill and activate position control, select zero parameter set, activate axis mode for the master spindle for spindle number <n>
SPCON: SPCON(<n>): SPCON(<n>,<m>):	Spindle position control ON for the master spindle for spindle number <n> for spindle numbers <n> and <m>
PCOF: SPCOF(<n>): SPCOF(<n>,<m>):	Spindle position control OFF, activate speed control mode for the master spindle for spindle number <n> for spindle numbers <n> and <m>
FPRAON(S<n>):	Revolutional feedrate for spindle <n> ON, derived from the master spindle
FPRAON(S<n>,S<m>):	Revolutional feedrate for spindle <n> ON, derived from spindle <m> The revolutional feedrate value must be specified with FA[S<m>].
FPRAOF(S<n>):	Revolutional feedrate for spindle <n> OFF

Statement	Description
C30 G90 G1 F3600	Rotary axis C (spindle in axis mode) travels to the position 30 degrees at a speed of 3600 degrees/min = 10 rpm
G25 S...: G25 S<n>:	Programmable minimum spindle speed limitation for the master spindle for spindle number <n>
G26 S...: G26 S<n>:	Programmable maximum spindle speed limitation for the master spindle for spindle number <n>
LIMS=...: LIMS[<n>]=...:	Programmable maximum spindle speed limitation with G96, G961, G97 for the master spindle for spindle number <n>
VELOLIM[<spindle>]=...:	Programmable limiting of the configured gear stage dependent maximum speed Using machine data (MD30455 $MA_MISC_FUNCTION_MASK, bit 6), when programming in the part program, it can be set as to whether VELOLIM is effective independent of whether used as spindle or axis (bit 6 = 1) - or is able to be programmed separately for each operating mode (bit 6 = 0). If they are to be separately effective, then the selection is made using the identifier when programming: • Spindle identifier S<n> for spindle operating modes • Axis identifier, e.g. "C", for axis operation The correction value refers to: • For spindles in axis operation (if MD30455 Bit 6 = 0): To the configured maximum axis velocity (MD32000 $MA_MAX_AX_VELO). • For spindles in spindle or axis operation (if MD30455 bit 6 = 1): To the maximum speed of the active gear unit stage (MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]) For further explanations about the programming of VELOIM, see: **References:** Programming Manual, Work Preparation
WAITS:	Synchronization command for master spindle The subsequent blocks are not processed until the spindle programmed in a previous NC block with SPOSA has reached its position (with exact stop fine). WAITS after M5: Wait until the spindle is stationary. WAITS after M3/M4: Wait for the spindle to reach its setpoint speed.
WAITS(<n>): WAITS(<n>,<m>):	Synchronization command for spindle number <n> Synchronization command for spindle numbers <n> and <m>
FA[S<n>]:	Programming of positioning speed (axial feed) for spindle <n> in [deg/min] With FA[S<n>]=0, the configured value takes effect once more: MD35300 $MA_SPIND_POSCTRL_VELO
OVRA[S<n>]:	Programming of the axial override value for spindle <n> in [%]
ACC[S<n>]:	Programming of the axial acceleration capacity of spindle <n> in [%]

Statement	Description
SPI(<n>):	With SPI(<n>) a spindle number is converted into the data type AXIS according to machine data MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[ ] . SPI is used if axis functions are to be programmed with the spindle. The following instructions are possible with SPI: • Frame instructions: – CTRANS() – CFINE() – CMIRROR() – CSCALE() • Velocity and acceleration values for following spindles: – FA[SPI(<n>)] – ACC[SPI(<n>)] – OVRA[SPI(<n>)] • System variables: – $P_PFRAME[SPI(<n>),TR]=<value> – $P_PFRAME= CTRANS(X,<axis value>,Y,<axis value>,SPI(<n>),<axis value>) – $P_PFRAME= CSCALE(X,<scale>,Y,<scale>,SPI(<n>),<scale>) – $P_PFRAME=CMIRROR(S<n>,Y,Z) – $P_UBFR=CTRANS(A,10) : CFINE (19,0.1) For further explanations about the programming of SPI, see: **References:** Programming Manual, Work Preparation.
M40: M<n>=40:	Automatic gear stage selection for the master spindle Automatic gear stage selection for spindle number <n>
M41 to M45: M<n>=41 to M<n>=45:	Select gear stage 1 to 5 for the master spindle Select gear stage 1 to 5 for spindle number <n>.

Note

M functions M3, M4, M5, and M70 are not output in DB21, ... DBB194 and DBB202 if a spindle is configured in a channel. These M functions are offered as extended M functions in DB21, ... DBB68 ff. and in the relevant axis DBs, DB31, ... DBB86 ff.

References

More detailed explanations for programming the spindle can be found in:

- Programming Manual, Fundamentals

15.7.2 Programming via synchronized actions

M functions M40 to M45 can also be programmed in synchronized actions.

Please note:

- The programming of M40 ... M45 in the part program has no effect on the current status of the automatic gear step change of synchronized actions, and vice versa.
- When programming S values with M40, automatic gear step change is effective separately for synchronized actions and the part program.
- M40 is deactivated after POWER ON.

  The gear step is not adjusted if an S value is specified from a synchronized action.
- An M40 command programmed using synchronized actions always remains active for synchronized actions (modal) and is not reset on reset.
- M41 ... M45 selects first to fifth gear steps in accordance with the programming in the part program.

  An axis replacement is necessary in order to run the function.

  Once the gear step change has been performed, the spindle status is neutral (same response to M3, M4, M5 programming).

References

For further explanations regarding the programming of the spindle as well as spindle movements from synchronized actions, refer to:

- Programming Manual, Job Planning
- Function Manual, Synchronized Actions

15.7.3 Programming spindle controls via PLC with FC18 - only 840D sl

When the PLC specifies the direction of rotation and speed using FC18, the NCK can determine and select a gear step to match the speed. This is equivalent to the M40 functionality when programming via the part program.

The correct start code must be set when FC18 is called in a PLC user program, in order to activate gear step selection.

References

More detailed explanations regarding the programming of spindle controls by PLC with FC18 can be found in:

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

15.7.4 Special spindle motion via the PLC interface

Note

The function is only available when using SINUMERIK Operate!

Why use a special spindle interface?

This function can be used to program the spindle via an axial PLC interface as an alternative to the FC18. The simplicity of the settings results in slightly restricted functionality. This functionality can be used preferably for simple control applications.

Functionality

Spindles can be started and stopped outside a part program that is being executed via the internal DBB30 spindle interface:

- DB31, … DBX30.0 (spindle stop)
- DB31, … DBX30.1 (spindle start, clockwise)
- DB31, … DBX30.2 (spindle start, counter-clockwise)
- DB31, ... DBX30.3 (select gear stage)

  Supplementary signal to the signal "spindle start, clockwise/counter-clockwise"; determines the gear stage that matches the speed analog to M40 in the part program.
- DB31, … DBX30.4 (spindle positioning start)

In order to start a spindle job, the channel handling the spindle must be in the acceptance state. A spindle job is always started on the Low-High edge of an internal DBB30 signal.

Generally, the internal DBB30 start signals do not have any meaning in the static state and do not prevent the spindle being programmed by FC18, synchronized actions, the part program or JOG traversing (e.g. when the STOP signal is statically at "1").

Requirements

In order that spindle jobs are accepted via the DBB30 spindle interface, the following conditions must be fulfilled (**acceptance state**):

- The channel state must be in the "interrupted" or "reset" mode:
  - DB21, ... DBX35.6 = 1 (channel state "interrupted")
  - DB21, ... DBX35.7 = 1 (channel state "reset")
- The program state must be in the "interrupted" or "canceled" mode:
  - DB21, ... DBX35.3 = 1 (program state "interrupted")
  - DB21, ... DBX35.4 = 1 (program state "interrupted")

These states will occur on RESET and in JOG mode.

At the start time, the spindle concerned must meet the following requirements:

- It must be in the state "Channel axis" or "Neutral axis" and must not be moved by means of the JOG keys.
- When the spindle is specified, no positioning may be carried out by FC18 or synchronized actions.

Note

Spindle job outside the acceptance range

Low-high edges outside the acceptance range will be ignored. No alarm message is output by the NCK. It can be conceivable that the acceptance range will be indicated to the operator by a user-side PLC application.

Spindle jobs outside the acceptance range can also be carried out using FC18 or ASUB.

Multi-channel operation

In the case of multi-channel operation, the spindle started by the PLC becomes active in the channel that handles the spindle at the appropriate moment.

The channel can be determined on the PLC side by reading the NC/PLC interface signals:

DB31, ... DBX68.0-68.3 (NC axis/spindle in channel A to D)

Spindle definitions

The spindle settings are retained after a change in mode (e.g. from JOG to AUTOMATIC). The spindle definitions are applied to the part program at the start of the program and can be modified again using part program operations.

Using the following settings in the machine data:
MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions),
certain spindle definitions (speed or cutting speed, feedrate type) can be taken from the part program, synchronized action and FC18 and entered in the corresponding setting data:

Bit 4 = 1	The programmed **speed** `S...` including speed setpoints via FC18 and synchronized actions are accepted in the following setting data: SD43200 $SA_SPIND_S (speed for spindle start via PLC interface) Programmed S values that are not programmed speed values are not accepted in the SD. These include, for example: • S value at constant cutting speed (G96, G961) • S value for rotation-related dwell time (G4)
Bit 8 = 1	The programmed **cutting speed** `S...` including setpoints via FC18 and synchronized actions are accepted in the following setting data: SD43202 $SA_SPIND_CONSTCUT_S (cutting speed for spindle start via PLC interface) Programmed S values that are not programmed cutting speed values are not accepted in the SD. These include, for example: • S value outside the constant cutting speed (G96, G961, G962) • S value for rotation-related dwell time (G4)
Bit 10 = 1	For the master spindle, the value of the 15th G group (**feedrate type**) is accepted in the following setting data SD43206 $SA_SPIND_SPEED_TYPE. SD43206 $SA_SPIND_SPEED_TYPE (spindle speed type for spindle start via PLC interface) For all other spindles, the value in SD43206 remains unchanged.

Speed default

Speed defaults from part program, FC18 or synchronized actions are written to the following setting data from all the usual sources:

SD43200 $SA_SPIND_S (speed for spindle start via PLC interface)

The setting data can be written to as follows:

- Through speed programming
- Through direct programming in the part program
- Through HMI software

Note

The setting data is written immediately and asynchronously to part-program execution.

The following conditions apply when writing:

Programming through:	Conditions:
Speed programming	• MD35035 $MA_SPIND_FUNCTION_MASK Bit 4 = 1 must be set. • Constant cutting speed G96, G961 must not be active. • The following NC/PLC interface signal must be set: DB31, ... DBX 84.0 = 0 (constant cutting speed is active)
Direct programming in the part program	A programmed S-value and the value of the directly written SD can be outdated with respect to time. If this is the case, after programming the SD, the statement `STOPRE` should be executed.
HMI	Only positive values including zero can be accepted. Otherwise, a corresponding message is generated.

Gear stage change and effect on speed

In the current version, no gear stage change is triggered if the setpoint speed is out of the speed range of the gear stage (exceptions, see "M40: Automatic gear stage selection for speeds outside the configured switching thresholds (Page 1489)"). The usual speed limitations and the speed increase to the setpoint speed are active.

Setpoint speed for JOG

For the following MD configuration:
MD35035 $MA_SPIND_FUNCTION_MASK bit 5 = 1
the content from SD43200 $SA_SPIND_S is active as setpoint speed for JOG.

You can use the JOG keys to operate the spindle at the speed defined in SD43200.

If the content is zero, other JOG speed definitions are active
(see SD41200 $SN_JOGSPIND_SET_VELO).

Constant cutting speed setting

Defaults for constant cutting speed from part program, FC18 or synchronized actions are written to the following setting data from all the usual sources:

SD43202 $SA_SPIND_CONSTCUT_S (cutting speed for spindle start via PLC interface)

Requirements

The requirement for the definition of a constant cutting speed to be active is:

- The spindle involved must be the master spindle in the channel that handles the spindle.

This condition is fulfilled if the following NC/PLC interface signal is set:

DB31, ... DBX84.0 = 1 (constant cutting speed is active)

Writing from the part program

When writing from the part program, the value for the constant cutting speed is interpreted as follows:

- If G710 is active in the 12th G group: Metric
- If G700 is set in the 12th G group: Inch as [feet/min]

For G70, G71 and when writing from an external (HMI), the setting in the machine data:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC
determines the interpretation of the written values.

For further explanations regarding the measuring system (metric/inch) see Section "G2: Velocities, setpoint / actual value systems, closed-loop control (Page 331)".

Definition via FC18

If the constant cutting speed is defined via FC18, the setting of bit 6 in byte 2 in the "Signals to concurring positioning axes" area determines how the speed value (bytes 8 ... 11) is interpreted.

Setting via synchronized actions

For definition via synchronized actions, analog to the part program, the feedrate type defines how the S value is interpreted.

Reading from part program and synchronized actions

The programmed cutting speed value can be determined in the part program and in the synchronized actions by reading the following system variables:

- $P_CONSTCUT_S[<n>] (last programmed constant cutting speed)
- $AC_CONSTCUT_S[<n>] (actual constant cutting speed)

Defined range of values of the two new system variables: RV = {0, DBL_Max}

The programmed cutting speed value can also be read via the OPI interface.

Definition of the spindle speed type for the master spindle

Definition of the spindle speed type **for the master spindle** from part program, FC18 or synchronized actions are written to the following setting data from all the usual sources:

SD43206 $SA_SPIND_SPEED_TYPE (spindle speed type for spindle start via PLC interface)

The value range and the functionality correspond to the 15th G group (feedrate type).

Permissible values are G values: 93, 94, 95, 96, 961, 97 and 971.

Depending on the setting, for DB31, … DBX30.1/2 (spindle start, clockwise/counter-clockwise) either the speed from SD43200 $SA_SPIND_S or the cutting speed from SD43202 $SA_SPIND_CONSTCUT_S is active:

93, 94, 95, 97 and 971:	The master spindle is started with the speed from SD43200.
96 and 961:	The speed of the master spindle is obtained from the specified cutting velocity (SD43202) and the radius of the transverse axis.

Definitions for spindle positioning

The definitions for spindle positioning using DB31, … DBX30.4 (spindle positioning start) are read from the following setting data:

SD43240 $SA_M19_SPOS (spindle position for spindle positioning with M19)

SD43250 $SA_M19_SPOSMODE (spindle position approach mode for spindle positioning with M19)

15.7.5 External programming (PLC, HMI)

SD43300 and SD42600

The revolutional feedrate behaviour can be selected externally via the axial setting data:
SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (Rotational feedrate for spindles)
in JOG operating mode using the channel-specific setting data
SD42600 $SC_JOG_FEED_PER_REV_SOURCE (Revolutional fedrate control 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:	Function is deselected.

FPRAON (S2)

Revolutional feedrate for spindle S2 ON, derived from the master spindle

FPRAON (S2, A)

Revolutional feedrate for spindle S2 ON, derived from axis A.
The revolutional feedrate value must be specified with `FA[Sn]`.

FPRAOF (S2)

Revolutional feedrate for spindle S2 OFF.

SPI(n)

It is also possible to program `SPI(n)` instead of `SPI(Sn)`.

15.8 Spindle monitoring

15.8.1 Permissible speed ranges

The permissible speed range of a spindle results from the parameterized or programmed speed limit values and the active spindle function (G94, G95, G96, G961, G97, G971, G33, G34, G35, G331, G332, etc.).

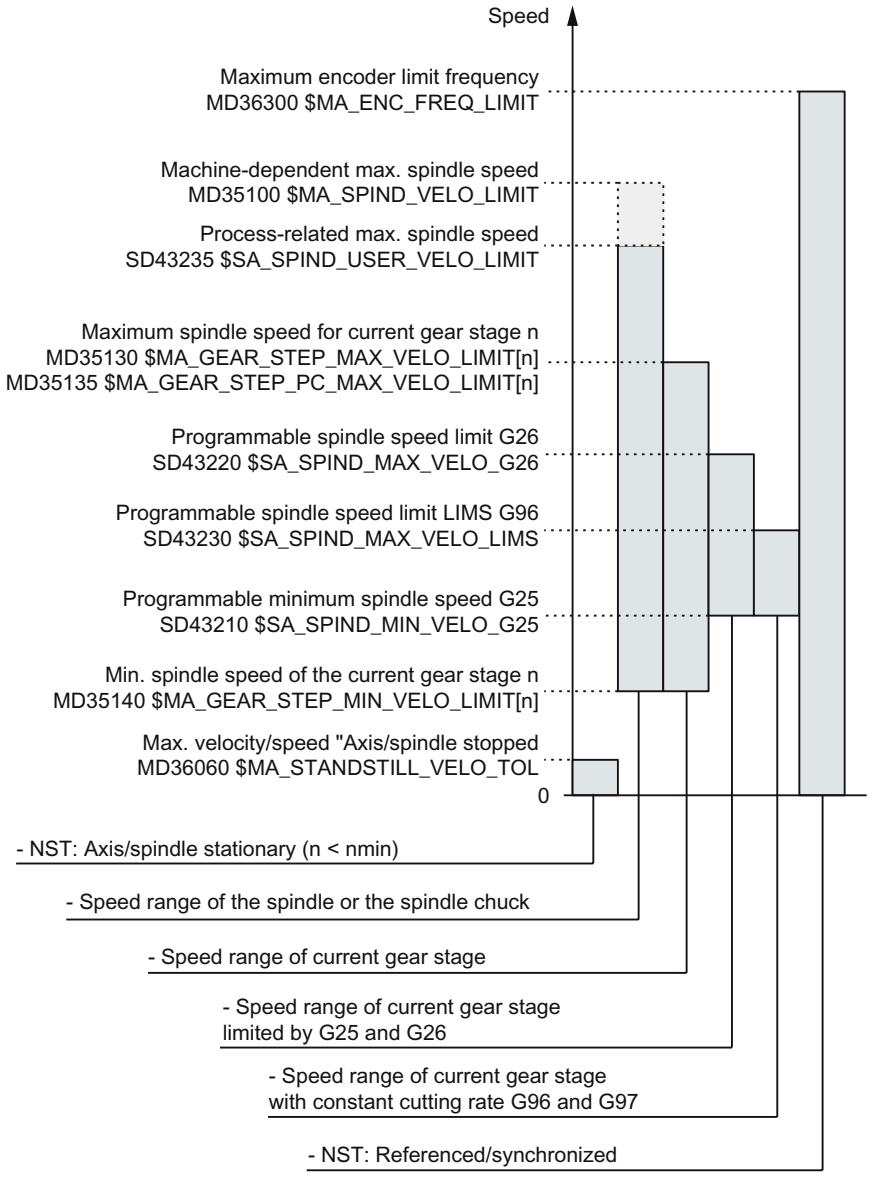


Figure 15-8 Ranges of spindle monitoring functions / speeds

15.8.2 Axis/spindle stationary

Functions such as tool change, open machine doors, path feedrate enable, etc. are only possible at the machine when the spindle is stationary.

Function

The "axis/spindle stationary" state is reached if a setpoint is no longer generated and the spindle actual speed falls below the configured threshold value for "axis/spindle stationary":

MD36060 $MA_STANDSTILL_VELO_TOL (max. velocity/speed for "axis/spindle stationary")

If the spindle has come to a standstill, the following NC/PLC interface signal is set:

DB31, ... DBX61.4 (axis/spindle stationary)

Effectiveness

Monitoring for spindle stop is effective in all spindle modes and in axis mode.

Deactivate path feed

If a spindle is stopped in the open-loop control mode (M5), then path feed is deactivated if the following machine data is set:

MD35510 $MA_SPIND_STOPPED_AT_IPO_START (feedrate enable for spindle stopped)

The path feed is re-enabled if the spindle comes to a standstill.

15.8.3 Spindle in setpoint range

Function

"Spindle in setpoint range" spindle monitoring checks whether:

- The programmed spindle speed is reached.
- The spindle is at a standstill:
  DB31, ... DBX61.4 (axis/spindle stationary) = 1
- The spindle is still in the acceleration or deceleration phase.

In the spindle mode, open-loop control mode, the setpoint speed is compared with the actual speed. If the actual speed deviates by more than the spindle tolerance that can be entered via MD (refer below) then:

- The following axial NC/PLC interface signal is set to "0":

  DB31, ... DBX83.5 (spindle in setpoint range) = 0

- The next machining block is not enabled (depending on the setting in MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START, see "Axis/spindle stationary (Page 1480)").

Spindle setpoint speed

The spindle setpoint speed is derived from the programmed speed taking into account the spindle correction and the active limits.

If the programmed speed is limited or increased, this is displayed using DB31, ... DBX83.1 (setpoint speed limited) or DB31, ... DBX83.2 (setpoint speed increased) (see also "Minimum / maximum speed of the gear stage (Page 1482)"). The means that reaching the tolerance range of the setpoint speed is **not** prevented.

Tolerance range for setpoint speed

The tolerance range of the setpoint speed is defined by the spindle speed tolerance factor:

MD35150 $MA_SPIND_DES_VELO_TOL

Example:

MD35150 $MA_SPIND_DES_VELO_TOL = 0.1

⇒ The spindle actual speed may deviate ±10% from the setpoint speed.

The following NC/PLC interface signal is set to "1" if the spindle actual speed lies within the tolerance range:

DB31, ... DBX83.5 (spindle in setpoint range) = 1

Special case:

If the spindle speed tolerance is set to "0", then DB31, ... DBX83.5 (spindle in the setpoint range) is permanently set to "1" and no path control is performed.

Speed change

Path control only takes place at the start of the traverse block and only if a speed change has been programmed. If the speed tolerance range is exited, e.g. due to an overload, the path movement is not automatically brought to a standstill.

15.8.4 Minimum / maximum speed of the gear stage

Minimum speed

The minimum speed of the gear stage of a spindle is configured in the machine data:

MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[<n>]

The speed setpoints, generated taking into account the override, do not fall below the minimum speed.

If an S value is programmed, which is less than the minimum speed, the setpoint speed is increased to the minimum speed and the following NC/PLC interface signal is set:

DB31, ... DBX83.2 (speed setpoint increased)

The minimum gear stage speed is effective only in speed mode and can only be undershot by:

- Spindle override 0%
- M5
- S0
- DB31, ... DBX4.3 (spindle stop)
- DB31, ... DBX2.1 (withdraw controller enable)
- DB21, ... DBX7.7 (reset)
- DB31, ... DBX2.2 (delete distance-to-go / spindle reset)
- DB31, ... DBX18.5 (oscillation speed)
- DB21, ... DBX7.4 (NC stop axes plus spindles)
- DB31, ... DBX1.3 (axis/spindle disable)
- DB31, ... DBX16.7 (delete S value)

Maximum speed

The maximum speed of the gear stage of a spindle is configured in machine data:

MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]

The speed setpoints, generated taking into account the override, are limited to this speed.

The following NC/PLC interface signal is set in the case that the speed is limited:

DB31, ... DBX83.1 (speed setpoint limited)

15.8.5 Diagnosis of spindle speed limitation

Function

The limit or increase of the spindle speed is signaled by the output of the following NC/PLC interface signals:

- DB31, ... DBX83.1 (setpoint speed limited)
- DB31, ... DBX83.2 (setpoint speed increased)

To diagnose the effective/limiting spindle parameters, one can have read access to the most important parameters of spindle motion via system variables. The system variables are indexed with the spindle number and they return only values that are relevant to the speed control and spindle position modes.

The following system variables are available in the spindle mode:

System variable	Meaning
$AC_SMAXVELO[<n>]	Maximum possible spindle speed [rpm] resulting from the active limiting data.
$AC_SMAXVELO_INFO[<n>]	Specification of the speed limiting data as numerical value. *)
$AC_SMINVELO[<n>]	Minimum possible spindle speed [rpm], corresponds to the minimum speed in the speed control mode.
$AC_SMINVELO_INFO[<n>]	Specification of the speed increasing data as numerical value. *)
$AC_SMAXACC[<n>]	Acceleration value of spindle [r/s^2].
$AC_SMAXACC_INFO[<n>]	Acceleration limiting cause in the form of a numerical value. *)
$AC_SPIND_STATE[<n>]	Status bits of the spindle.
<n>: Spindle no. (n= 0: the variables are related to the current master spindle)	

*) The text of the numerical value should be taken from from the description of the system variables.

Evaluation of diagnosis data:
The system variables for each spindle can be read via synchronized actions and in the part program, giving due consideration to the preprocessing stop in the NCK .

Boundary conditions

The values delivered by the system variables depend on the spindle mode:

- Speed control mode:

  All system variables deliver current values.

- Positioning mode:

  The system variables $AC_SMAXVELO, $AC_SMAXACC and $AC_SPIND_STATE deliver valid values. The system variables $AC_SMINVELO and $AC_SMINVELO_INFO deliver the data that becomes effective on changing to the speed control mode.

- Axis mode (e.g. if the spindle is used by a transformation TRANSMIT, TRACYL,... or follows a path motion as a special axis):

  The system variable $AC_SPIND_STATE can also be used in the axis mode. Separate system variables are available in the axis mode for dynamic data:

  $AA_VMAXM, $AA_VMAXB and $AA_VLFCT.

The following control response is obtained for a type SERUPRO block search:

- The system variable $AC_SMAXVELO / $AC_SMAXACC delivers the maximum representable speed / acceleration.
- $AC_SMAXVELO_INFO and $AC_SMAXACC_INFO deliver the VALUE "0" (no limitation is active).
- $AC_SMINVELO and $AC_SMINVELO_INFO deliver data as in case of normal part program processing.
- $AC_SPIND_STATE returns the states as they are set for SERUPRO.

Example

Example of the visualization of the content of the system variables for Spindle 1. The variables are written to the R parameters cyclically. These can be displayed on HMI in the R Parameters area.

```
Program code
N05 IDS=1 WHENEVER TRUE DO $R10=$AC_SMAXVELO[1]
N10 IDS=2 WHENEVER TRUE DO $R11=$AC_SMAXVELO_INFO[1]
N15 IDS=3 WHENEVER TRUE DO $R12=$AC_SMINVELO[1]
N20 IDS=4 WHENEVER TRUE DO $R13=$AC_SMINVELO_INFO[1]
N25 IDS=5 WHENEVER TRUE DO $R14=$AC_SPIND_STATE[1]
```

See also

Spindle in setpoint range (Page 1480)

15.8.6 Maximum spindle speed

Machine-related parameterizable maximum spindle speed

The machine-related maximum spindle speed is parameterized via the following machine data:

MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

The speed limit value is monitored by the NC in the actual values, i.e. taking into account the current gear stage.

Note

Machine manufacturer

It is recommended that the reduction of the maximum spindle speed in machine data MD35100 only be performed when the spindle is stationary. This is particularly important for spindles that are active after NC reset (see machine data description of MD35040 SPIND_ACTIVE_AFTER_RESET). Otherwise alarm 22100 may be output.

References

- Manufacturer documentation: List Manual, "Detailed Machine Data Description"
- User documentation: Diagnostics Manual

Responses to violation

If the actual speed of the spindle exceeds the parameterized maximum spindle speed by more than the tolerance value (MD35150 $MA_SPIND_DES_VELO_TOL), this results in the following responses on the NC side:

- DB31, ... DBX83.0 = 1 (speed limit exceeded)
- Alarm 22100 "Chuck speed exceeded" is displayed
- All axes and spindles of the channel are stopped

Process-related settable maximum spindle speed

Using the following setting data, it is possible to set a lower maximum spindle speed in relation to machine data MD35100 because of the process:

SD43235 $SA_SPIND_USER_VELO_LIMIT (maximum spindle speed)

A change can be made via:

- Programming in the part program
- The user interface by the machine operator

The change becomes active immediately.

Limitation of the speed setpoint

The controller limits the spindle speed setpoint to the value specified in the setting data.

A limitation of the spindle speed setpoint due to setting data SD43235 is displayed via the following system variable with the value "21":

$AC_SMAXVELO_INFO[<spindle number>] == 21

15.8.7 Maximum encoder limit frequency

⚠CAUTION

The maximum encoder frequency limit of the actual spindle position encoder is monitored by the control (the limit can be exceeded). It is the responsibility of the machine tool manufacturer to ensure that the configuration of the spindle motor, gearbox, measuring gearbox, encoder and machine data prevents the maximum speed of the actual spindle position encoder being exceeded.

Maximum encoder frequency exceeded.

If the spindle speed reaches a speed (large S value programmed), which exceeds the maximum encoder limit frequency (the maximum mechanical speed limit of the encoder must not be exceeded), the synchronization is lost. The spindle continues to rotate, but with reduced functionality.

With the following functions, the spindle speed is reduced until the active measurement system is operating below the encoder limit frequency again:

- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck (G331, G332)
- Revolutional feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- SPCON (position-controlled spindle operation)

When the encoder limit frequency is exceeded
NC/PLC IS:
DB31, ... DBX60.4 (referenced/synchronized 1)
or
DB31, ... DBX60.5 (referenced/synchronized 2)
are reset for the measurement system in question and NC/PLC IS:
DB31, ... DBX60.2 (encoder limit frequency 1 exceeded)
or
DB31, ... DBX60.3 (encoder limit frequency 2 exceeded)
are set.

If the spindle is in axis mode, the maximum encoder limit frequency must not be exceeded. The maximum velocity (MD32000 $MA_MAX_AX_VELO) must lie below the maximum encoder limit frequency; otherwise, alarm 21610 is output and the axis is brought to a standstill.

Maximum encoder limit frequency unundershot

If the maximum encoder frequency limit has been exceeded and the speed subsequently falls below the maximum encoder limit frequency (smaller S value programmed, spindle offset switch changed, etc.), the spindle is automatically synchronized with the next zero mark or the next Bero signal. The new synchronization will always be carried out for the active position measuring system that has lost its synchronization and whose max. encoder limit frequency is currently undershot.

Special features

If the following functions are active, the maximum encoder frequency cannot be exceeded:

- Spindle positioning mode, axis mode
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck G331, G332 (does not apply to G63)
- Revolutional feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- SPCON

15.8.8 End point monitoring

End point monitoring

During positioning (the spindle is in positioning mode), the system monitors the distance from the spindle (with reference to the actual position) to the programmed spindle position setpoint (end point).

For this to work, in machine data:
MD36000 $MA_STOP_LIMIT_COARSE (Exact stop limit coarse)
and
MD36010 $MA_STOP_LIMIT_FINE (Exact stop limit fine)
two limit values can be defined as an incremental path starting from the spindle position setpoint.

Regardless of the two limit values, the positioning of the spindle is always as accurate as defined by the connected spindle measurement encoder, the backlash, the transmission ratio, etc.

Exact stop window dependent on parameter set

Various parameter-set-dependent exact stop windows can be configured.
This makes it possible to work to different levels of accuracy in axis mode and spindle positioning. The exact stop window can be configured separately for each gear step for spindle positioning.

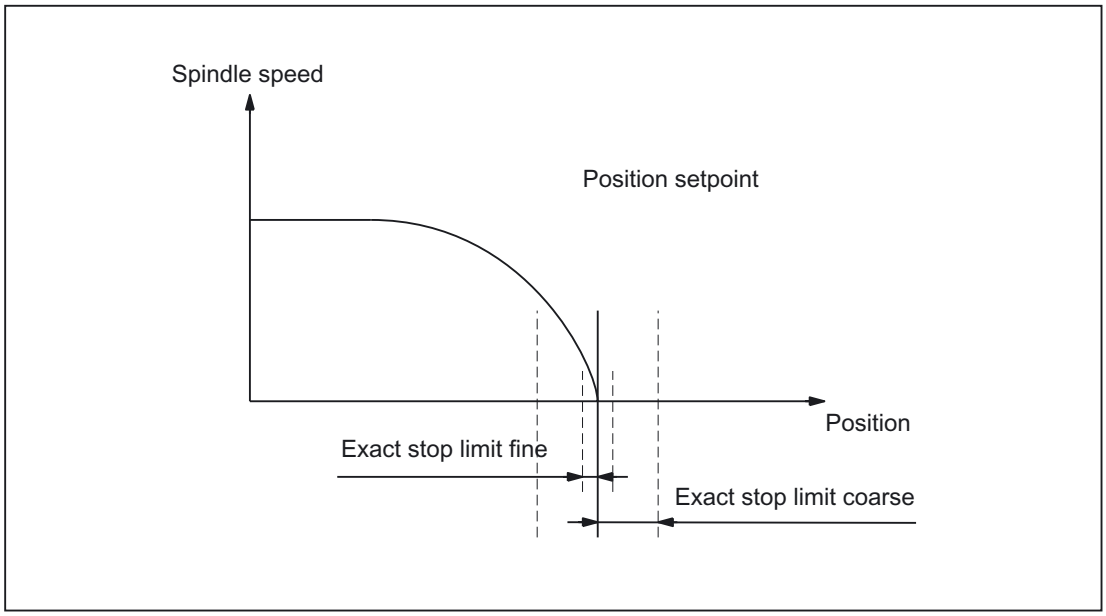


Figure 15-9 Exact stop zones of a spindle

DB31, ... DBX60.7 and DB31, ... DBX60.6 (position reached with exact stop coarse / fine)

The two limit values defined by machine data:
MD36000 $MA_STOP_LIMIT_COARSE (Exact stop limit coarse)
and
MD36010 $MA_STOP_LIMIT_FINE (Exact stop limit fine)
are output to the PLC using NC/PLC IS:
DB31, ... DBX60.7 (Position reached with exact stop coarse)
and
DB31, ... DBX60.6 (Position reached with exact stop fine).

Block change for SPOS and M19

When positioning the spindle with SPOS or M19 the block is changed dependent on end point monitoring with NC/PLC IS:
DB31, ... DBX60.6 (Position reached with exact stop fine).

All other functions programmed in the block must have achieved their end criterion (e.g., all auxiliary functions acknowledged by the PLC).

With SPOSA, the block change does not depend on the monitoring of the end point.

15.8.9 M40: Automatic gear stage selection for speeds outside the configured switching thresholds

Function

When M40 is active, an automatic gear stage selection is also made if the programmed spindle speed S… lies outside the configured switching thresholds.

In this case, a distinction is made between the following cases:

- **Programmed speed too high**

  The programmed speed is higher than the configured maximum speed of the numerically largest gear stage:

  S... > MD35110 $MA_GEAR_STEP_MAX_VELO[<n>]

  In this case, the **highest gear stage** is selected (according to MD35090 $MA_NUM_GEAR_STEPS).

- **Programmed speed too low**

  The programmed spindle speed is less than the configured minimum speed of the first gear stage:

  S... < MD35120 $MA_GEAR_STEP_MIN_VELO[1]

  In this case, the **first gear stage** is selected.

- **Programmed speed = 0**

  When programming speed 0 (S0) the behavior depends on the configuration of the minimum speed of the first gear stage MD35120 $MA_GEAR_STEP_MIN_VELO[1]:

  – If MD35120 $MA_GEAR_STEP_MIN_VELO[1] **= 0** is configured, then when programming S0, the **first gear stage** is selected.

  – If MD35120 $MA_GEAR_STEP_MIN_VELO[1] **> 0** is configured, when programming S0 **no gear stage change** is performed and the last gear stage remains active. This means that it remains possible to stop the spindle with S0 (instead of M5) without initiating a gear stage change.

Effectiveness

Selecting the highest gear stage or the first gear stage for automatic gear stage selection (M40) is active when programming spindle speeds S… using the part program, in synchronized actions or when entering via PLC FC18.

For speed programming from the part program for tapping with G331, the behavior is also supported for the second data set to select the gear stage (precondition: MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, bit 5 = 1).

Boundary conditions

Enabling the gear stage change

The precondition for the function is that the gear stage change is generally enabled via machine data:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear stage change)

MD35090 $MA_NUM_GEAR_STEPS (number of gear stages set up)

Example

Automatic gear stage selection M40 is the initial setting after NC reset.

Part program:

```
Program code              Comment
...
N15 S3500 M3              ; S3500 is greater than MD35110 of the 2nd gear stage. The
                            2nd gear stage is selected.
...
N50 S0 M3                 ; Spindle is stopped, S0 does not request a gear stage
                            change (special handling, S0).
...
N95 S5 M3                 ; S5 is less than MD35120 of the 1st gear stage. The 1st
                            gear stage is selected.
...
```

Configuring data for spindle 1 (AX5):

```
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE[AX5] = 1    ; Enable gear stage change
MD35090 $MA_NUM_GEAR_STEPS[AX5] = 2             ; Number of existing gear stages
MD35110 $MA_GEAR_STEP_MAX_VELO[1,AX5] = 500     ; Upper switching threshold for gear stage 1
MD35120 $MA_GEAR_STEP_MIN_VELO[1,AX5] = 10      ; Lower switching threshold for gear stage 1
MD35110 $MA_GEAR_STEP_MAX_VELO[2,AX5] = 2000    ; Upper switching threshold for gear stage 2
MD35120 $MA_GEAR_STEP_MIN_VELO[2,AX5] = 500     ; Lower switching threshold for gear stage 2
```

15.9 Spindle with SMI 24 (Weiss spindle)

15.9.1 General Information

In order to be able to process the sensor data of the spindle in the control, the sensors must first be connected to I/O modules and transferred to the PLC via fieldbus (PROFIBUS DP or PROFINET I/O).

For a spindle with SMI 24 (Weiss spindle), the sensor data are transferred to the drive using DRIVE-CLiQ and are available there in drive parameters. When using cyclic drive telegram 139, sensor data from the drive are transferred to the control. They are then available there in the following system data:

- System variable
- OPI variables
- NC/PLC interface signals

Requirement

- The spindle is connected to the drive via Sensor Module SMI 24 using DRIVE-CLiQ.
- Drive telegram 139 is configured for the spindle.

Note

Drive telegram 139

In principle, a spindle with Sensor Module SMI 24 can also be operated with another drive telegram. However, sensor data is only transferred to the control using drive telegram 139.

15.9.2 Sensor data

Sensors in the spindle motor

The spindle sensors provide information about the clamping device and the angular position of the motor shaft:

- Analog sensor S1: Clamped state

  Voltage value in the range from 0 - 10 V depending on the position of the draw bar.
- Digital sensor S4: Piston end position
  - 0 = piston not in position
  - 1 = piston is in position, i.e. piston is free to move
- Digital sensor S5: Angular position of the motor shaft
  - 0 = motor shaft not aligned
  - 1 = motor shaft is in position (requirement: The spindle is stationary)

Note

Spindle with sensor module SMI 24 and axis container

A spindle with sensor module SMI 24 and drive telegram 139 for the transmission of sensor data to the control must not be part of an axis container whose axes are distributed over several NCUs via an NCU link.

Transmission of sensor data

Sensor data are transferred to the control from sensor module SMI 24 in cyclic drive telegram 139 as process data 11 - 14. Drive telegram 139 is based on drive telegram 136, where sensor data are transferred instead of the data of the 2nd encoder. A detailed description of drive telegram 139 can be found in:

References:
SINAMICS S120/S150 List Manual; Function Diagrams, Section: PROFIdrive

System data: Sensor data

Sensor data can be read into the control via the following system data:

Meaning	System variable $VA_	NC/PLC interface DB31, ...	OPI variables	Drive parameters
Sensor configuration	MOT_SENSOR_CONF[<axis>]	DBB132	vaMotSensorConf	r5000
Clamped state 1)	MOT_CLAMPING_STATE[<axis>]	DBW134	vaMotClampingState	r5001
Measured value analog sensor S1 2)	MOT_SENSOR_ANA[<axis>]	DBW136	vaMotSensorAna	r5002
Status digital sensors	MOT_SENSOR_DIGI[<axis>]	DBW138	vaMotSensorDigi	r5003
<axis>:	Machine axis identifier: AX1 ... AXn or spindle identifier: S1 ... Sm			

1) See Section "Clamped state (Page 1495)"

2) Sensor S1: 0 - 10 V
Analog actual value: 0 - 10000 increments, resolution 1 mV
Example:
SIMATIC S7 input module: 0 - 27648 increments, resolution 0.36 mV
Adaptation factor if you change to a spindle with SMI 24: 2,7648

Detailed system data description

For information about NC/PLC interfaces, see Section "Signals from axis/spindle (DB31, ...) (Page 1900)".

References:

System variable:	List Manual, System Variables
OPI variables:	List Manual 2; Variables
Drive parameters:	SINAMICS S120/S150 List Manual

15.9.3 Clamped state

Sensor S1 supplies an analog voltage value 0 V - 10 V depending on the position of the clamping device. The voltage value is available in the system data for evaluation of the clamped state on the user side.

Note

The subsequently described evaluation of sensor S1 to generate the state value for the clamped state and limiting the spindle speed are only realized if the following state values are displayed in drive parameter r5000:

- r5000.0 == 1: Sensors available
- r5000.1 == 1: Sensor S1 (clamped state) available
- r5000.10 == 1: State values are generated, speed limits p5043 active

See also Section "Sensor data (Page 1493)", paragraph: "System data: sensor data"

State value

To simplify the evaluation, the clamped state in the system data is also available as state value 0 - 11.

A voltage range corresponds to a certain clamped state. The voltage ranges can be set using drive parameter p5041[0...5].

A voltage tolerance can also be set for the voltage ranges using drive parameter p5040.

Note

The voltage range ± voltage tolerance must not overlap.

Speed limits

For the clamped states with state values 3 - 10, speed limit values can be specified using drive parameter p5043[0...6]. In the other clamped states (state values 1, 2 and 11), a limit value of 0 [rpm] permanently applies.

In the various clamped states, the controller limits the spindle speed to the applicable limit.

Note

Changing the speed limits

A change of the speed limits in drive parameter p5043[0...6] is only effective in the controller (limitation of the spindle speed to the new speed limit) after:

- Power-on reset or when the controller is switched-off/switched-on
- Deselection of the "Parking" state for the spindle (see Section "Deactivating all monitoring functions: "Parking" (Page 123)")

Context: State value, voltage range and speed limit

State value [1]	Clamped state	Voltage range [2]		Speed limit
		Upper limit	Lower limit	
0	Sensor S1 not available or state values inactive	-	-	-
1	State initialization running	-	-	[3]
2	Released with signal (error state)	-	p5041[0] + p5040	[3]
3	Released	p5041[0]	p5041[1]	p5043[0]
4	Clamping with tool	-	-	p5043[1]
5	Releasing with tool	-	-	p5043[2]
6	Releasing without tool	-	-	p5043[3]
7	Clamped with tool AND S4 == 0	p5041[2]	p5041[3]	p5043[4]
8	Clamped with tool AND S4 == 1			p5043[4]
9	Clamping without tool	-	-	p5043[5]
10	Clamped without tool	p5041[4]	p5041[5]	p5043[6]
11	Clamped with signal (error state)	p5041[5] - p5040	-	[3]

[1] The state value can be read into the controller using the following system data:

- System variable: $VA_MOT_CLAMPING_STATE[<axis>]
- NC/PLC interface: DB31, ... DBW134
- OPI variables: vaMotClampingState
- Drive parameters: r5001

[2] p5041[0...5]: Voltage threshold values, p5040: Voltage threshold values tolerance

[3] Speed limit permanently set: 0 [rpm]

15.9.4 Additional drive parameters

P5042:Transition time

The following times can be set in drive parameter p5042 for the clamped state identification:

- p5042[0]: Stabilization time for "clamped with tool"
  The clamped state "clamped with tool" must be present in the spindle motor for at least the set stabilization time before the state is signaled to the controller.
- p5042[1]: Maximum time for clamping
  The transition from the "released" state to the "clamped with tool" or "clamped without tool" state may take – as a maximum – the set time.

r5044: Speed limitation from the clamping cycle

The speed limit from p5043[6] which is active in the clamped state "clamped without tool" is displayed in drive parameter r5044.

A value of 65535 means that the speed limit is not active.

15.10 Supplementary conditions

15.10.1 Changing control parameters

For spindles that are not in position-controlled mode, machine data changes also take effect when the spindle is not stationary with the NEWCONF command.

In the case of changes to control parameters, speed setpoint jumps may occur when the new values take effect. Control parameters are, for example:

- MD32200 $MA_POSCTRL_GAIN (servo gain factor)
- MD32210 $MA_POSCTRL_INTEGR_TIME (position controller integral time)
- MD32410 $MA_AX_JERK_TIME (time constant for the axial jerk filter)

15.11 Examples

15.11.1 Automatic gear step selection (M40)

Example

To illustrate the contents of the new block search variables:
Assumptions about automatic gear step selection (M40):

S0...500	1. Gear step
S501..1000	2. Gear step
S1001..2000	3. Gear step

Content of system variables:

$P_SEARCH_S	; Collected S value
$P_SEARCH_DIR	; Collected direction of rotation
$P_SEARCH_GEAR	; Collected gear step

Collected	S value:	Direction of rotation:	Gear step:
	; 0/last speed	-5	40/last GS
N05 G94 M40 M3 S1000	; 1000	3	40
N10 G96 S222	: 222	3	40
N20 G97	; f (PlanAxPosPCS)*	3	40
N30 S1500	; 1500	3	40
N40 SPOS=0**	; 1500	-19	40
N50 M19**	; 1500	-19	40
N60 G94 G331 Z10 S300	; 300	-19	40
N70 M42	; 300	-19	42
N80 M4	; 300	4	42
N90 M70	; 300	70	42
N100 M3 M40	; 300	3	40
N999 M30			

* f (PlanAxPosPCS): The speed depends on the current position of the transverse axis in the workpiece coordinate system.

** ($P_SEARCH_SPOS and $P_SEARCH_SPOSMODE are programmed)

15.12 Data lists

15.12.1 Machine data

15.12.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10192	GEAR_CHANGE_WAIT_TIME	Wait time for acknowledgment of a gear stage change during reorganization
10714	M_NO_FCT_EOP	M function for spindle active after RESET
12060	OVR_SPIND_IS_GRAY_CODE	Spindle override with Gray coding
12070	OVR_FACTOR_SPIND_SPEED	Evaluation of spindle speed override switch
12080	OVR_REFERENCE_IS_PROG_FEED	Override reference velocity
12082	OVR_REFERENCE_IS_MIN_FEED	Defining the reference of the path override
12090	OVR_FUNCTION_MASK	Selection of override specifications

15.12.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20090	SPIND_DEF_MASTER_SPIND	Initial setting for master spindle on channel
20092	SPIND_ASSIGN_TAB_ENABLE	Enabling/disabling of spindle converter
20850	SPOS_TO_VDI	Output of M19 to the PLC with SPOS/SPOSA
22400	S_VALUES_ACTIVE_AFTER_RESET	S function active after RESET

15.12.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis
30310	ROT_IS_MODULO	Modulo conversion
31044	ENC_IS_DIRECT2	Encoder on intermediate gear
31050	DRIVE_AX_RATIO_DENOM	Denominator load gearbox
31060	DRIVE_AX_RATIO_NUMERA	Numerator load gearbox
31064	DRIVE_AX_RATIO2_DENOM	Intermediate gear denominator
31066	DRIVE_AX_RATIO2_NUMERA	Intermediate gear numerator
31070	DRIVE_ENC_RATIO_DENOM	Measuring gear denominator
31080	DRIVE_ENC_RATIO_NUMERA	Measuring gear numerator
31122	BERO_DELAY_TIME_PLUS	BERO delay time in plus direction
31123	BERO_DELAY_TIME_MINUS	BERO delay time in minus direction
32200	POSCTRL_GAIN	K_V factor
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant current control circuit for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control circuit for feedforward control
32910	DYN_MATCH_TIME	Time constant for dynamic matching
34040	REFP_VELO_SEARCH_MARKER	Reference point creep speed
34060	REFP_MAX_MARKER_DIST	Monitoring of zero mark distance
34080	REFP_MOVE_DIST	Reference point distance/destination point for distancecoded system
34090	REFP_MOVE_DIST_CORR	Reference point offset/absolute offset, distancecoded
34100	REFP_SET_POS	Reference point value
34200	ENC_REFP_MODE	Homing mode
35000	SPIND_ASSIGN_TO_MACHAX	Assignment of spindle to machine axis
35010	GEAR_STEP_CHANGE_ENABLE	Type of gear step change
35012	GEAR_STEP_CHANGE_POSITION	Gear step change position
35014	GEAR_STEP_USED_IN_AXISMODE	Gear step for axis mode with M70
35020	SPIND_DEFAULT_MODE	Basic spindle setting
35030	SPIND_DEFAULT_ACT_MASK	Activate initial spindle setting
35035	SPIND_FUNCTION_MASK	Setting of spindle-specific functions
35040	SPIND_ACTIVE_AFTER_RESET	Spindle active after reset
35090	NUM_GEAR_STEPS	Number of installed gear steps
35092	NUM_GEAR_STEPS2	2nd gear step data set: Number of installed gear steps
35100	SPIND_VELO_LIMIT	Maximum spindle speed
35110	GEAR_STEP_MAX_VELO[n]	Maximum speed for automatic gear stage change

Number	Identifier: $MA_	Description
35112	GEAR_STEP_MAX_VELO2[n]	2nd gear step data set: Maximum speed for automatic gear stage change
35120	GEAR_STEP_MIN_VELO[n]	Minimum speed for automatic gear stage change
35122	GEAR_STEP_MIN_VELO2[n]	2nd gear step data set: Minimum speed for automatic gear stage change
35130	GEAR_STEP_MAX_VELO_LIMIT[n]	Maximum speed of gear step
35135	GEAR_STEP_PC_MAX_VELO_LIMIT[n]	Maximum speed of gear step in position control
35140	GEAR_STEP_MIN_VELO_LIMIT[n]	Minimum speed of gear step
35150	SPIND_DES_VELO_TOL	Spindle speed tolerance
35160	SPIND_EXTERN_VELO_LIMIT	Spindle speed limitation via PLC
35200	GEAR_STEP_SPEEDCTRL_ACCEL[n]	Acceleration in speed control mode
35210	GEAR_STEP_POSCTRL_ACCEL[n]	Acceleration in position control mode
35212	GEAR_STEP_POSCTRL_ACCEL2[n]	2nd gear step data set: Acceleration in position control mode
35220	ACCEL_REDUCTION_SPEED_POINT	Speed limit for reduced acceleration
35230	ACCEL_REDUCTION_FACTOR	Reduced acceleration
35300	SPIND_POSCTRL_VELO	position control activation speed
35350	SPIND_POSITIONING_DIR	Positioning direction of rotation for a nonsynchronized spindle
35400	SPIND_OSCILL_DES_VELO	Oscillation speed
35410	SPIND_OSCILL_ACCEL	Oscillation acceleration
35430	SPIND_OSCILL_START_DIR	Oscillation start direction
35440	SPIND_OSCILL_TIME_CW	Oscillation time for M3 direction
35450	SPIND_OSCILL_TIME_CCW	Oscillation time for M4 direction
35500	SPIND_ON_SPEED_AT_IPO_START	Feed enable with spindle in setpoint range
35510	SPIND_STOPPED_AT_IPO_START	Feed enable with stationary spindle
35550	DRILL_VELO_LIMIT[n]	Maximum speeds for tapping
35590	PARAMSET_CHANGE_ENABLE	Parameter set definition possible from PLC
36060	STANDSTILL_VELO_TOL	Threshold velocity "Axis/spindle stationary"
36200	AX_VELO_LIMIT	Threshold value for velocity monitoring.

15.12.2 Setting data

15.12.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42600	JOG_FEED_PER_REF_SOURCE	Revolutional feedrate control in JOG mode
42800	SPIND_ASSIGN_TAB	Spindle number converter
42900	MIRROR_TOOL_LENGTH	Mirror tool length offset
42910	MIRROR_TOOL_WEAR	Mirror wear values of tool length compensation
42920	WEAR_SIGN_CUTPOS	Mirror wear values of machining plane
42930	WEAR_SIGN	Invert sign of all wear values
42940	TOOL_LENGTH_CONST	Retain the assignment of tool length components when changing the machining plane (G17 to G19)

15.12.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43200	SPIND_S	Speed for spindle start via PLC interface
43202	SPIND_CONSTCUT_S	Cutting rate for spindle start via PLC interface
43206	SPIND_SPEED_TYPE	For spindle start via PLC interface
43210	SPIND_MIN_VELO_G25	Progr. Spindle speed limiting G25
43220	SPIND_MAX_VELO_G26	Progr. Spindle speed limiting G26
43230	SPIND_MAX_VELO_LIMS	Progr. spindle speed limiting G96/G961
43235	SPIND_USER_VELO_LIMIT	Maximum spindle speed
43240	M19_SPOS	Spindle position for spindle positioning with M19
43250	M19_SPOSMODE	Spindle positioning approach mode for spindle positioning with M19
43300	ASSIGN_FEED_PER_REF_SOURCE	Rotational feedrate for positioning axes/spindles

15.12.3 signals

15.12.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate override A to H	DB31, ...DBX0.0-7	DB380x.DBX0.0-7
Axis/spindle disable	DB31, ...DBX1.3	DB380x.DBX1.3
Followup mode	DB31, ...DBX1.4	DB380x.DBX1.4
Position measuring system 1	DB31, ...DBX1.5	DB380x.DBX1.5
Position measuring system 2	DB31, ...DBX1.6	DB380x.DBX1.6
Override active	DB31, ...DBX1.7	DB380x.DBX1.7
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Spindle reset/delete distancetogo	DB31, ...DBX2.2	DB380x.DBX2.2
Velocity/spindle speed limitation	DB31, ...DBX3.6	DB380x.DBX3.6
Program test Axis/Spindle Enable	DB31, ...DBX3.7	DB380x.DBX3.7
Actual gear step A to C	DB31, ...DBX16.0-2	DB380x.DBX2000.0-2
Gear changed	DB31, ...DBX16.3	DB380x.DBX2000.3
Resynchronize spindle 1	DB31, ...DBX16.4	DB380x.DBX2000.4
Resynchronize spindle 2	DB31, ...DBX16.5	DB380x.DBX2000.5
no n-monitoring with gear change	DB31, ...DBX16.6	DB380x.DBX2000.6
Delete S value	DB31, ...DBX16.7	DB380x.DBX2000.7
Feedrate override for spindle valid	DB31, ...DBX17.0	DB380x.DBX2001.0
Resynchronize spindle during positioning 1	DB31, ...DBX17.4	DB380x.DBX2001.4
Resynchronize spindle during positioning 2	DB31, ...DBX17.5	-
Invert M3/M4	DB31, ...DBX17.6	DB380x.DBX2001.6
Oscillation via PLC	DB31, ...DBX18.4	DB380x.DBX2002.4
Oscillation enable (oscillation speed)	DB31, ...DBX18.5	DB380x.DBX2002.5
Oscillation rotation direction clockwise (Set rotation direction clockwise)	DB31, ...DBX18.6	DB380x.DBX2002.6
Oscillation rotation direction counterclockwise (Set rotation direction counterclockwise)	DB31, ...DBX18.7	DB380x.DBX2002.7
Spindle override A to H	DB31, ...DBX19.0-7	DB380x.DBX2003.0-7

15.12.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Spindle/No Axis	DB31, ...DBX60.0	DB390x.DBX0.0
Encoder limit frequency exceeded 1	DB31, ...DBX60.2	DB390x.DBX0.2
Encoder limit frequency exceeded 2	DB31, ...DBX60.3	-
Homed/synchronized 1	DB31, ...DBX60.4	DB390x.DBX0.4
Homed/synchronized 2	DB31, ...DBX60.5	DB390x.DBX0.5
Position reached with exact stop coarse	DB31, ...DBX60.6	DB390x.DBX0.6
Position reached with exact stop fine	DB31, ...DBX60.7	DB390x.DBX0.7
Axis/spindle stationary ($n < n_{min}$)	DB31, ...DBX61.4	DB390x.DBX1.4
Position controller active	DB31, ...DBX61.5	DB390x.DBX1.5
Speed controller active	DB31, ...DBX61.6	DB390x.DBX1.6
Current controller active	DB31, ...DBX61.7	DB390x.DBX1.7
Restored 1	DB31, ...DBX71.4	-
Restored 2	DB31, ...DBX71.5	-
Setpoint gear stage A to C	DB31, ...DBX82.0-2	DB390x.DBX2000.0-2
Change gear	DB31, ...DBX82.3	DB390x.DBX2000.3
Speed limit exceeded	DB31, ...DBX83.0	DB390x.DBX2001.0
Setpoint speed limited	DB31, ...DBX83.1	DB390x.DBX2001.1
Setpoint speed increased	DB31, ...DBX83.2	DB390x.DBX2001.2
Spindle in setpoint range	DB31, ...DBX83.5	DB390x.DBX2001.5
Actual direction of rotation clockwise	DB31, ...DBX83.7	DB390x.DBX2001.7
Rigid tapping active	DB31, ...DBX84.3	DB390x.DBX2002.3
active spindle mode synchronous mode	DB31, ...DBX84.4	DB390x.DBX2002.4
Active spindle positioning mode	DB31, ...DBX84.5	DB390x.DBX2002.5
Active spindle mode oscillation mode	DB31, ...DBX84.6	DB390x.DBX2002.6
Active spindle control mode	DB31, ...DBX84.7	DB390x.DBX2002.7
Spindle actually reached in position	DB31, ...DBX85.5	DB390x.DBX2003.5
M function for spindle	DB31, ...DBB86-87	DB370x.DBD0000
S function for spindle	DB31, ...DBB88-91	DB370x.DBD0004
Sensors available	DB31, ...DBX132.0	DB390x.DBX7000.0
Sensor S1 available (clamped state)	DB31, ...DBX132.1	DB390x.DBX7000.1
Sensor S4 available (piston end position)	DB31, ...DBX132.4	DB390x.DBX7000.4
Sensor S5 available (angular position of the motor shaft)	DB31, ...DBX132.5	DB390x.DBX7000.5
State value is generated, speed limitation p5043 is active	DB31, ...DBX133.2	DB390x.DBX7001.2
Clamped state	DB31, ...DBW134	DB390x.DBW7002
Analog value: Clamped state	DB31, ...DBW136	DB390x.DBW7004
Sensor S4 (piston end position)	DB31, ...DBX138.4	DB390x.DBX7006.4
Sensor S5 (angular position of the motor shaft)	DB31, ...DBX138.5	DB390x.DBX7006.5

V1: Feedrates

16

16.1 Brief description

Types of feedrate

The feedrate determines the machining speed (axis or path velocity) and is observed in every type of interpolation, even where allowance is made for tool offsets on the contour or on the tool center point path (depending on G functions).

The following types of feedrate allow optimum adaptation to the various technological applications (turning, milling, drilling, etc.):

- Rapid traverse feedrate (G0)
- Inverse-time feedrate (G93)
- Linear feedrate (G94)
- Revolutional feedrate (G95)
- Constant cutting rate (G96, G961)
- Constant speed (G97, G971)
- Feedrate for thread cutting (G33, G34, G35)
- Feedrate for tapping without compensating chuck (G331, G332)
- Feedrate for tapping with compensating chuck (G63)
- Feedrate for chamfer/rounding FRC, FRCM
- Non-modal feedrate FB

Axis assignment of the feedrates

Feedrates can be assigned to the axes variably to adjust to the different technological requirements.

The following versions are possible:

- Separate feedrates for the working plane and the infeed axis
- Variable axis assignment for path feedrate
- Feedrate for positioning axes

Feedrate control

The programmed feedrate can be changed during the machining or for test purposes to enable adjustment to the changed technological conditions.

- Via the machine control panel
- Via the operator panel front
- Via the PLC
- Per program command

16.2 Path feedrate F

Path feedrate F

The path feedrate represents the geometrical total of the speed components in the participating axes. It is therefore generated from the individual motions of the interpolating axes.

The default uses the axial speeds of the geometry axes which have been programmed. The `FGROUP` command can be used to include other geometry and/or synchronized axes in the calculation of the path feedrate.

The path feedrate F determines the machining speed and is observed in every type of interpolation even where allowance is made for tool offsets. The value programmed under the address F remains in a program until a new F value or a new type of feedrate is programmed.

Range of values for path feedrate F

See Description of Functions G2: "Speeds, Setpoint / Actual Value Systems, Closed-Loop Control", Section: "Velocities (Page 332)".

F value at PLC interface

The F value of the current path feedrate is always entered in the channel-specific PLC interface for auxiliary functions (DB21, ... DBB158 to 193).

For explanations about the associated interface signals (change signal, F value), see Description of Functions "H2: Auxiliary function outputs to PLC (Page 395)".

Feedrate with transition circle

References:
Programming Manual, Fundamentals

Feedrate for internal radius and external radius path sections

For circular blocks or spline blocks with curvature in the same direction and tool radius offset activated (G41/G42), the programmed feedrate can act on the center point path or on the contour (depending on the internal radius or external radius path sections).

A group of G functions is provided for this purpose:

- CFTCP

  Programmed feedrate acting on the center point path.

- CFC

  Programmed feedrate acting on the contour.

- CFCIN

  Programmed feedrate acting only on the contour with a concave spline.

References:
Programming Manual, Fundamentals

Maximum tool path velocity

The maximum path velocity results from the maximum velocities of the linear or rotary axes involved (MD32000 $MA_MAX_AX_VELO), i.e. the axis with the lowest maximum velocity determines the maximum path velocity. This cannot be exceeded.

If `G0` is programmed, traversing is at the path velocity resulting from the MD32000 $MA_MAX_AX_VELO limitation.

Limit velocity for path axes

In addition, the `FL[<axis>]` statement can be used to program a limit velocity for path axes (geometry and synchronized axes).

This enables separate feedrates to be programmed for the working plane and infeed axis. This means that a feedrate is specified for the path-related interpolation and for the infeed axis. The axis perpendicular to the selected machining plane is designated as the infeed axis. The infeed axis-specific feedrate can be programmed to limit the axis velocity and therefore the path velocity. No coordinate rotations through frames should be included, i.e. the infeed axis must be an axis of the standard coordinate system. This function can be used to compensate for the fact that a cutter has a lower cutting performance on the face side than across the cutter circumference.

Programming example:

Program code	Comment
`... G94 ...`	`; Selection of feedrate type (mm/min)`
`X30 Y20 F200`	`; Path feedrate = 200 mm/s`
`FL[Z]=50 Z-30`	`; Max. feedrate for Z axis: 50 mm/s`

Low-resolution encoders

When using low-resolution encoders, more continuous path or axis motions can be achieved with smoothed actual values. The larger the time constant, the better the smoothing of the actual values, and the longer the overtravel.

MD34990 $MA_ENC_ACTVAL_SMOOTH_TIME[<axis>] (smoothing time constant for actual values)

Smoothed actual values are used for:

- Thread cutting (G33, G34, G35)
- Feedrate per revolution ((G95, G96, G97, FPRAON)
- Display of speed, actual position and velocity

16.2.1 Feedrate type G93, G94, G95

Effectiveness

The feedrate types G93, G94, G95 are active for the G functions of group 1 (except G0) in the automatic modes.

G94 or G95 can be used for traversing in JOG mode.

References:
Function Manual, Extended Functions; Manual and Handwheel Travel (H1)

Inverse-time feedrate (G93)

The inverse-time feedrate is used when it is easier to program the duration, rather than the feedrate, for retraction of a block.

The inverse-time feedrate is calculated from the following formula:

$$F = \frac{v}{s}$$

with F: Inverse-time feedrate in rpm
v: Required path velocity in mm/min or inch/min
s: Path length in mm/inch

Programming example:

Program code	Comment
N10 G1 G93 X100 Y200 F2	; The programmed path is traversed in 0.5 min.
...	

Note

G93 may not be used when G41/G42 is active. If the block length varies greatly from block to block, a new F value should be programmed in each block for G93.

Linear feedrate (G94)

The linear feedrate is programmed in the following units relative to a linear or rotary axis:

- [mm/min, degrees/min] on standard metric systems
- [inch/min, degrees/min] on standard imperial systems

Revolutional feedrate (G95)

The revolutional feedrate is programmed in the following units relative to a master spindle:

- [mm/rev] on standard metric systems
- [inch/rev] on standard imperial systems
- [degrees/rev] on a rotary axis

The path velocity is calculated from the actual speed of the spindle according to the following formula:

$$V = n * F$$

with V: Path velocity in mm/min or inch/min
n: Speed of the master spindle in rpm
F: Programmed revolutional feedrate in mm/rev or inch/rev

Note

The programmed F value is deleted when the system switches between the feedrate types G93, G94 and G95.

Tooth feedrate

Primarily for milling operations, the tooth feedrate FZ... (feed distance per tooth), which is more commonly used in practice, can be programmed instead of the revolutional feedrate F...:

The controller uses the $TC_DPNT (number of teeth per revolution) tool parameter associated with the active tool offset data record to calculate the effective revolutional feedrate for each traversing block from the programmed tooth feedrate.

$$F = FZ * \$TC\_DPNT$$

with F: Revolutional feedrate in mm/rev or inch/rev
FZ: Tooth feedrate in mm/tooth or inch/tooth
$TC_DPNT: Tool parameter: Number of teeth/rev

Example: Milling cutter with 5 teeth ($TC_DPNE = 5)

```
Program code                Comment
N10 G0 X100 Y50
N20 G1 G95 FZ=0.02          ; Tooth feedrate 0.02 mm/tooth
N30 T3 D1                   ; Load tool and activate tool offset data record.
M40 M3 S200                 ; Spindle speed 200 rpm
N50 X20                     ; Milling with:
                              FZ = 0.02 mm/tooth
                              → effective revolutional feedrate:
                              F = 0.02 mm/tooth * 5 teeth/rev = 0.1 mm/rev
                              or
                              F = 0.1 mm/rev * 200 rpm = 20 mm/min
…
```

Revolutional feedrate in JOG mode

In JOG mode, the response of the axis/spindle also depends on the following setting data:

SD41100 $SN_JOG_REV_IS_ACTIVE (revolutional feedrate for JOG active)

If this setting data is active, an axis/spindle is always moved with revolutional feedrate:
MD32050 $MA_JOG_REV_VELO (revolutional feedrate with JOG)
or
MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feedrate with JOG with rapid traverse overlay)
depending on the master spindle.

If the setting data is inactive:

- The response of the axis/spindle depends on the setting data:

  SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (revolutional feedrate for position axes/spindles)

- The response of a geometry axis on which a frame acts is to rotate, depending on the channel-specific setting data:

  SD42600 $SC_JOG_FEED_PER_REV_SOURCE

DB31, ... DBX62.2 (revolutional feedrate active)

A programmed, active revolutional feedrate (G95) is displayed using the following NC/PLC interface signal.

DB31, ... DBX62.2 (revolutional feedrate active)

16.2.2 Type of feedrate G96, G961, G962, G97, G971

Constant cutting rate (G96, G961)

The constant cutting rate is used on turning machines to keep the cutting conditions constant, independently of the work diameter of the workpiece. This allows the tool to be operated in the optimum cutting performance range and therefore increases its service life.

Selection of G96, G961:

When programming G96, G961, the corresponding S value is interpreted as the cutting rate in m/min or ft/min along the transverse axis. If the workpiece diameter decreases during machining, the speed is increased until the constant cutting speed is reached.

When G96, G961 is first selected in the part program, a constant cutting rate must be entered in mm/min or ft/min.

With G96, the control system will automatically switch to revolutional feedrate (as with G95), i.e. the programmed feedrate F is interpreted in mm/rev or inch/rev.

When programming G961, linear feedrate is selected automatically (as with G94). A programmed feedrate F is interpreted in mm/min or inch/min.

Determining the spindle speed

Based on the programmed cutting rate (either S_{G96} or S_{G961}) and the actual cartesian position of the transverse axis (radius), the control system calculates the spindle speed at the TCP using the following formula:

$$n = \frac{S_{Speed}}{2 * \pi * r}$$

n: Spindle speed

S_{Speed}: Programmed cutting rate

π Circle constant

r: Radius (distance, center of rotation to TCP)

The following is assumed when determining the radius:

- The transverse axis position 0 in the WCS represents the center of rotation.
- Position offsets (such as online tool offset, external zero offset, $AA_OFF, DRF offset and compile cycles) and position components through couplings (e.g. following axis for TRAIL) are not taken into account when determining the radius.

Frames (e.g. programmable frames such as SCALE, TRANS or ROT) are taken into account in the calculation of the spindle speed and can bring about a change in speed, if the effective diameter at the TCP changes.

Diameter programming and reference axis for several transverse axes in one channel:

One or more transverse axes are permitted and can be activated simultaneously or separately:

- Programming and displaying in the user interface in the diameter
- Assignment of the specified reference axis with `SCC[<axis>]` for a constant cutting rate G96, G961, G962

For more information, see Description of Functions "P1: Transverse axes (Page 893)".

Example

S_{G96} = 230 m/min

- Where r = 0.2 m → n = 183.12 rpm
- Where r = 0.1 m → n = 366.24 rpm

⇒ The smaller the workpiece diameter, the higher the speed.

For G96, G961 or G962, a geometry axis must be defined as the transverse axis.

The transverse axis, whose position affects the speed of the mater spindle, is defined using channel-specific machine data:

MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function)

The function G96, G961 or G962 requires that the machine zero and the workpiece zero of the transverse axis are in the turning center of the spindle.

Constant speed (G97, G971)

G97, G971 deactivates the "Constant cutting rate function" (G96, G961) and saves the last calculated spindle speed. With G97, the feedrate is interpreted as a revolutional feedrate (as with G95).

When programming G971, linear feedrate is selected (as with G94). The feedrate F is interpreted in mm/min or inch/min.

When G97, G971 is active, an S value can be reprogrammed to define a new spindle speed. This will not modify the cutting rate programmed in G96, G961.

G97, G971 can be used to avoid speed variations in motions along the transverse axis without machining (e.g. cutting tool).

Note

G96, G961 is only active during workpiece machining (G1, G2, G3, spline interpolation, etc., where feedrate F is active).

The response of the spindle speed for active G96, G961 and G0 blocks can be defined in the channelspecific machine data:

MD20750 ALLOW_G0_IN_G96 (G0 logic for G96, G961)

When constant cutting rate G96, G961 is selected, no gear stage change can take place.

The spindle speed override switch acts on the spindle speed calculated.

A DRF offset in the transverse axis does not affect the spindle speed setpoint calculation.

At the start of machining (after G0) and after NC stop, G60, G09, ... the path start waits for "nAct= nSet".

The interface signals "nAct = nSet" and "Set speed limited" are not modified by internal speed settings.

When the speed falls below the minimum speed or if the signal "Axis/spindle stationary" is recognized, "nAct =nSet" is reset.

A path operation which has started (G64, rounding), is not interrupted.

Spindle speed limitation with G96, G961

A maximum spindle speed can be specified for the "Constant cutting rate" function:

- In the setting data:

  SD43230 $SA_SPIND_MAX_VELO_LIMS (spindle speed limitation for G96/G961)

- In the part program (for the master spindle) with the programming command LIMS

The most recently changed value (LIMS or SD) is active.
LIMS is effective with G96, G961, G97 and can be specified on up to four speed limitations in the part program in one block. Spindle number <Sn> = 1, 2, 3, or 4 of the master spindle that is possible in the particular instance can be programmed in part program instruction LM[<Sn>].

Note

When the block is loaded in the main run, all programmed values are transferred to the setting data SD43230 $SA_SPIND_MAX_VELO_LIMS.

Depending on the machine data:
MD10710 PROG_SD_RESET_SAVE_TAB[n] (setting data to be updated),
the speed limit set with LIMS remains stored after the controller is switched off.
When G96, G961, G97 are reactivated, this speed limitation is also activated.

The maximum permissible spindle speed defined via G26 or via the setting data:
SD43220 $SA_SPIND_MAX_VELO_G26 (maximum spindle speed)
cannot be exceeded.

In the event of incorrect programming that would cause one of the speed limits (G26 or SD43220 $SA_SPIND_MAX_VELO_G26) to be exceeded, the following interface signal is set.

DB31, ... DBX83.1 (programmed speed too high)

In order to ensure smooth rotation with large part diameters, the spindle speed is not permitted to fall below a minimum level.
This speed can be set via the setting data:
SD43210 $SA_SPIND_MIN_VELO_G25 (minimum spindle speed)
and, depending on the gear step, with the machine data:
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT (minimum speed of the gear step)
.

The minimum spindle speed can be changed in the part program with G25. In the event of programming that would mean one of the spindle limits (G25 or SD43220 $SA_SPIND_MAX_VELO_G25) is not reached, the following interface signal is set.

DB31, ... DBX83.2 (speed setpoint too low)

For detailed information on the spindle speed limitations and the mode of operation of the setting data, see Description of Functions S1: "Spindles", Section: "Spindle monitoring (Page 1479)".

Master spindle switchover with G96, G961

If the master spindle is switched over when G96, G961 are active, the speed of the former master spindle is retained. This corresponds to a transition from G96 to G97. The master spindle newly defined with SETMS executes the "Constant cutting rate" function generated in this way.

Alarms

Constant cutting rate G96, G961, G962

- If no F value is programmed, alarm 10860 "No feedrate programmed" is output. The alarm is not generated with G0 blocks.
- Alarm 14800 "Programmed path velocity smaller than or equal to zero" is output while programming a negative path velocity.
- If, with an active G96, G961 or G962, no transverse axis is defined in the machine data: MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function), alarm 10870 "No transverse axis defined" is output.
- If a negative maximum spindle speed is programmed with the `LIMS` program command when G96, G961 are active, alarm 14820 "Negative maximum spindle speed programmed for G96, G961" is output.
- If no constant cutting rate is programmed when G96, G961 is selected for the first time, alarm 10900 "No S value programmed for constant cutting rate" is output.

16.2.3 Feedrate for thread cutting (G33, G34, G35)

G33

The function G33 can be used to machine threads with constant pitch of the following type:

Speed S, feedrate F, thread pitch

A revolutional feedrate [mm/revolution] is used for G33 threads. The revolutional feedrate is defined by programming the thread pitch [mm/revolution].

The speed of the axes for the thread length is calculated from the programmed spindle speed S and the thread pitch.

Feedrate F [mm/min] = speed S [rev/min] * pitch [mm/rev]

At the end of the acceleration ramp, the position coupling between the spindle actual value (spindle setpoint with SPCON on master spindle) and the axis setpoint is established. At this moment, the position of the axis in relation to the zero mark of the spindle (including zero mark offsets) is as if the axis had accelerated abruptly at the start of the block when the thread start position (zero mark plus SF) was crossed. Compensation is made for the following error of the axis.

Minimum spindle speed

In order to ensure smooth rotation at low speeds, the spindle speed is not permitted to fall below a minimum level.

This speed can be set:

- With the setting data:

  SD43210 $SA_SPIND_MIN_VELO_G25 (minimum spindle speed)

- For each gear stage with the machine data:

  MD35140 $MA_GEAR _STEP_MIN_VELO_LIMIT (minimum speed for gear stage change)

The minimum spindle speed can be changed in the part program with G25.

NC stop, single block

NC stop and single block (even at the block boundary) are only active after completion of thread chaining. All successive G33 blocks and the first following non-G33 block are traversed as a block.

Premature abortion without destruction

Thread cutting can be aborted without destruction before the end point is reached. This can be done by activating a retraction motion.

Thread cutting with ROT frame

With ROT frame and G33, G34, G35, alarm 10607 "Thread with frame not executable" is activated if the rotation causes a change in the thread length and thus the pitch. Rotation around the thread axis is permissible.

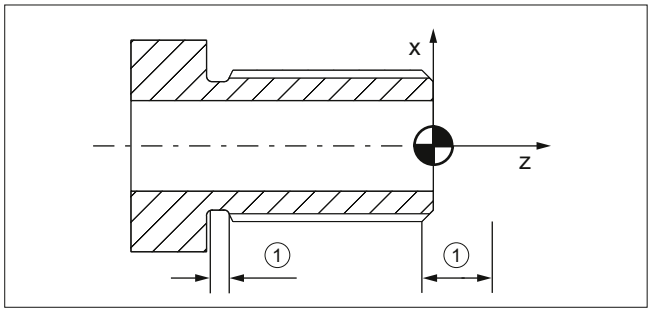
Alarm 10607 "Thread with frame not executable" can be suppressed by setting bit 12 in machine data MD11410 $MN_SUPPRESS_ALARM_MASK, if the ROT instruction is used intentionally in the application.

All other frames are accepted by the NC without alarm. Attention is drawn to the pitch-changing effect of SCALE.

16.2.3.1 Programmable run-in and run-out path for G33, G34 and G35

Function

The run-in and run-out path of the thread can be specified with the DITS and DITE statements. The thread axis is accelerated or braked within the specified path.



① Run-in/run-out path, depending on the machining direction

Short run-in path

Due to the collar on the thread run-in, little room is left for the tool start ramp.
This must therefore be specified shorter via DITS.

Short run-out path

Due to the collar on the thread run-out, little room is left for the deceleration ramp, leading to the risk of collision between the workpiece and the tool edge. The deceleration ramp can be specified shorter using DITE. Due to the inertia of the mechanical system, however, a collision can still occur.

Remedy: Program a shorter thread, reduce the spindle speed.

Note

DITE acts at the end of the thread as a rounding clearance. This achieves a smooth change in the axis motion.

Effects

The programmed run-in and run-out path only increases the rate of acceleration on the path. If one of the two paths is set larger than the thread axis needs with active acceleration, the thread axis is accelerated or decelerated with maximum acceleration.

Note

The axis can be overloaded if the specified path is too short.

Activation

The DITS and DITE statements are always active for thread cutting.

Example

```
Program code                                    Comment
N...
N59 G90 G0 Z100 X10 SOFT M3 S500
N60 G33 Z50 K5 SF=180 DITS=1 DITE=3             ; Start of corner rounding with Z=53
N61 G0 X20
```

Note

Only paths, and not positions, are programmed with DITS and DITE.

The programmed run-in/run-out path is handled according to the current dimension setting (inches, metric).

Setting data SD42010 (acceleration behavior of axis when thread cutting)

When a block containing DITS and/or DITE is inserted in the main run, the programmed run-in/run-out path is taken into the setting data:

- SD42010 $SC_THREAD_RAMP_DISP[**0**] = programmed value of DITS
- SD42010 $SC_THREAD_RAMP_DISP[**1**] = programmed value of DITE

If no run-in/run-out path is programmed before or in the first thread block, the current value of the setting data is used.

SD42010 $SC_THREAD_RAMP_DISP[**0**] = <value>

SD42010 $SC_THREAD_RAMP_DISP[**1**] = <value>

<value>	Meaning
0 > <value> ≥ -1	The feed axis is accelerated with the acceleration according to the current programming BRISK/SOFT
<Value> = 0	The feed axis is accelerated in steps (BRISK).
<Value> > 0	The maximum thread run-up or deceleration distance is specified. **Note** Too short a distance can result in an acceleration overload of the axis.

With reset / program end, the setting data is reset to the values -1, -1.

MD10710 $MN_PROG_SD_RESET_SAVE_TAB can be used to specify that the value of the setting data written with DITS and DITE can be saved retentively at reset / program end and therefore retained after power on.

16.2.3.2 Linear increasing/decreasing thread pitch change with G34 and G35

Function

The thread pitch increase (G34) defines the numerical increase in the pitch value. A larger pitch results in a larger distance between the threads on the workpieces. The velocity of the thread axis therefore increases with assumed constant spindle speed.

The opposite therefore applies for the decrease in thread pitch (G35).

The following definitions are made for the thread pitch change:

- G34: Increase in thread pitch corresponds to progressive change
- G35: Decrease in thread pitch corresponds to degressive change

Both G34 and G35 functions imply the functionality of G33 and also provide the option of programming an absolute pitch change value for the thread under F. If the start and end pitch of a thread is known, the thread pitch change can be determined using the following equation:

$$F = \frac{|k_e^2 - k_a^2|}{2 * l_G}$$

The meaning is as follows:

F: The thread pitch change to be programmed [mm/rev^2]

k_e: Thread pitch of axis target point coordinate, thread axis [mm/rev]

k_a: Initial thread pitch (programmed under I, J or K) [mm/rev]

l_G: Thread length [mm]

The absolute value of F must be applied to G34 or G35 depending on the required pitch increase of decrease.

When the thread length l_G, pitch change F and initial pitch k_a are known, the pitch increase at the end of block k_e can be determined as follows by modifying the formula:

- For G34 (increasing pitch):

$$k_e = \sqrt{k_a^2 + F * 2 * l_G}$$

- For G35 (decreasing pitch):

$$k_e = \sqrt{k_a^2 - F * 2 * l_G}$$

Note

If the formula results in a negative root expression, the thread cannot be machined!

In this case, the NC signals alarm 10605 or alarm 22275.

Application

The G34 and G35 functions can be used to produce self-shearing threads.

Example

Thread cutting G33 with decreasing thread pitch G35

```
Program code                          Comment
N1608 M3 S10                          ; Spindle speed
N1609 G0 G64 Z40 X216                 ; Approach starting point
N1610 G33 Z0 K100 SF=R14              ; Thread with constant pitch 100 mm/rev
N1611 G35 Z-220 K100 F17.045455       ; Thread pitch decrease 17.045455 mm/rev2
                                      ; Thread pitch at end of block 50 mm/rev
N1612 G33 Z-240 K50                   ; Traverse thread block without jerk
N1613 G0 X218
N1614 G0 Z40
N1616 M17
```

Monitoring during the block preparation

Any pitch changes that would overload the thread axis when G34 is active or would result in an axis standstill when G35 is active, are detected in advance during block preparation. Alarm 10604 "Thread pitch increase too high" or 10605 "Thread pitch decrease too high" is signaled.

During thread cutting, certain practical applications require a correction of the spindle speed. In this case, the operator will base his correction on the permissible velocity of the thread axis.

To do this, it is possible to suppress the output of alarms 10604 and 10605 as follows:

MD11410 $MN_SUPPRESS_ALARM_MASK bit 10 = 1

Block preparation is then continued normally.

Monitoring during the execution

The following situations are monitored cyclically when the thread is machined (interpolation):

- Exceeding of maximum velocity of thread axis
- Reaching of axis standstill with G35

The following alarm is signaled when the monitoring function responds:

- Alarm 22269 "Maximum velocity of thread axis reached" or
- Alarm 22275 "Zero velocity of thread axis reached"

16.2.3.3 Fast retraction during thread cutting

Function

The "Fast retraction for thread cutting (G33)" function can be used to interrupt thread cutting without causing irreparable damage in the following circumstances:

- NC stop (NC/PLC interface signal)
- Alarms that implicitly trigger NC stop
- Switching of a rapid input
  References
  Programming Manual, Job Planning; Section "Fast retraction from the contour"

The retraction motion can be programmed via:

- Retraction path and retraction direction (relative)
- Retraction position (absolute)

Note

Tapping

The "Fast retraction" function **cannot** be used with tapping (G331/G332).

Programming

Syntax

Enable fast retraction, retraction motion via retraction path and retraction direction:
```
G33 ... LFON DILF=<value> LFTXT/LFWP ALF=<value>
```

Enable fast retraction, retraction motion via retraction position:

```
POLF[<axis identifier>]=<value> LFPOS
POLFMASK/POLFMLIN(<axis 1 name>,<axis 2 name>, etc.)
G33 ... LFON
```

Disable fast retraction for thread cutting:
```
LFOF
```

Meaning

LFON:	Enable fast retraction for thread cutting (G33).
LFOF:	Disable fast retraction for thread cutting (G33).
DILF= :	Define length of retraction path. The value preset during MD configuration (MD21200 $MC_LIFTFAST_DIST) can be modified in the part program by programming DILF. **Note:** The configured MD value is always active following NC-RESET.
LFTXT LFWP:	The retraction direction is controlled in conjunction with ALF with G functions LFTXT and LFWP. LFTXT: The plane in which the retraction motion is executed is calculated from the path tangent and the tool direction (default setting). LFWP: The plane in which the retraction motion is executed is the active working plane.
ALF= :	The direction is programmed in discrete degree increments with ALF in the plane of the retraction motion. With LFTXT, retraction in the tool direction is defined for ALF=1. With LFWP the direction in the working plane is derived from the following assignment: • G17 (X/Y plane) ALF=1 ; Retraction in the X direction ALF=3 ; Retraction in the Y direction • G18 (Z/X plane) ALF=1 ; Retraction in the Z direction ALF=3 ; Retraction in the X direction • G19 (Y/Z plane) ALF=1 ; Retraction in the Y direction ALF=3 ; Retraction in the Z direction

References:
Programming options with ALF are also described in "Traverse direction for fast retraction from the contour" in the Programming Manual, Job Planning.

LFPOS: Retraction of the axis declared with POLFMASK or POLFMLIN to the absolute axis position programmed with POLF.

POLFMASK: Release of axes (<axis 1 name>,<axis 1 name>, etc.) for independent retraction to absolute position.

POLFMLIN: Release of axes for retraction to absolute position in linear relation

Note:
Depending on the dynamic response of all the axes involved, the linear relation cannot always be established before the lift position is reached.

POLF[]: Define absolute retraction position for the geometry axis or machine axis in the index

Effectiveness: Modal

=<value>: In the case of geometry axes, the assigned value is interpreted as a position in the workpiece coordinate system. In the case of machine axes, it is interpreted as a position in the machine coordinate system.

The values assigned can also be programmed as incremental dimensions:
=IC<value>

<axis identifier>: Identifier of a geometry axis or machine axis.

Note

LFON or LFOF can always be programmed, but the evaluation is performed exclusively during thread cutting (G33).

Note

POLF with POLFMASK/POLFMLIN are not restricted to thread cutting applications.

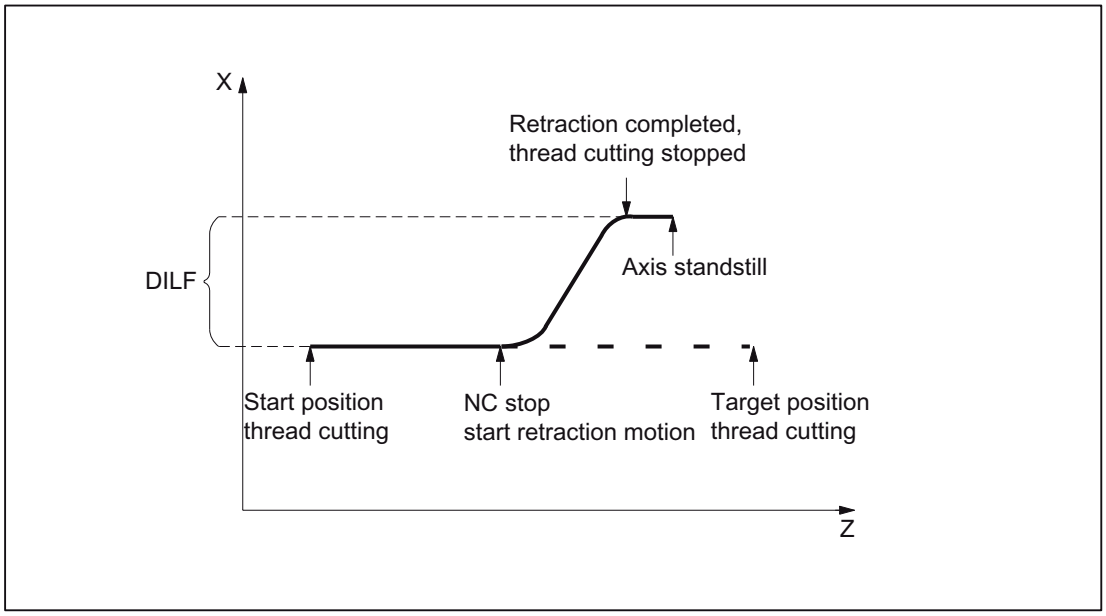


Figure 16-1 Interruption of G33 through retraction motion

Dynamic response of the retraction motion

The retraction motion is executed with maximum axis dynamic response:

- MD32000 $MA_MAX_AX_VELO[<axis>] (velocity)
- MD32300 $MA_MAX_AX_ACCEL[<axis>] (acceleration)
- MD32431 $MA_MAX_AX_JERK[<axis>] (jerk)

Example

```
Program code                                    Comment
N55 M3 S500 G90 G18                             ; Set active machining plane.
...
N65 MSG ("thread cutting")
MM_THREAD:
N67 $AC_LIFTFAST=0                              ; Reset before starting the thread.
N68 G0 Z5
N69 X10
N70 G33 Z30 K5 LFON DILF=10 LFWP ALF=7          ; Enable fast retraction for thread cutting.
                                                ; Retraction path = 10 mm
                                                ; Retraction plane Z/X (because of G18).
                                                ; Retraction direction -X (with ALF=3; retraction
                                                  direction +X).
N71 G33 Z55 X15
N72 G1                                          ; Deselect thread cutting.
N69 IF $AC_LIFTFAST GOTOB MM_THREAD             ; If thread cutting has been interrupted.
N90 MSG ("")
...
N70 M30
N55 M3 S500 G90 G0 X0 Z0
...
N87 MSG ("tapping")
N88 LFOF                                        ; Deactivate fast retraction before tapping.
N89 CYCLE...                                    ; Tapping cycle with G33.
N90 MSG ("")
...
N99 M30
```

Behavior at power on and reset

After power on and reset, the following settings are activated:

- Initial settings for the retraction motion (LFON /LFOF) and retraction direction (LFTXT/LFWP): MD20150 $MC_GCODE_RESET_VALUES
- Retraction path: MD21200 $MC_LIFTFAST_DIST

16.2.4 Feedrate for tapping without compensating chuck (G331, G332)

Function

A thread can be tapped without compensating chuck with the functions G331 (tapping) and G332 (tapping retraction).

Requirement

The technical requirement for tapping without compensating chuck is a position-controlled spindle with position measuring system.

Speed S, feedrate F, thread pitch

A revolutional feedrate [mm/rev] is used for G331 and G332. The revolutional feedrate is defined by programming the thread pitch [mm/rev].

The speed of the axes for the thread length is calculated from the programmed spindle speed S and the thread pitch.

Feedrate F [mm/min] = speed S [rev/min] * pitch [mm/rev]

Override

The revolutional feedrate in G331 and G332 can be influenced by an override.

Depending on the configuration, the override affects either the spindle speed or the path feedrate:

MD12090 $MN_OVR_FUNCTION_MASK

Bit	Value	Meaning
0	0	The override influences the spindle speed (initial setting). Depending on the setting in the machine data: MD12080 $MN_OVR_REFERENCE_IS_PROG_FEED the override is related either to the programmed spindle speed or to the configured spindle speed limitation.
	1	The override influences the path feedrate Depending on the setting in machine data: MD12082 $MN_OVR_REFERENCE_IS_MIN_FEED the override is related either to the programmed path feedrate or to the configured path feedrate limitation.

Note

The following overrides are not effective in G331 and G332:

- Programmable path feedrate override OVR
- Rapid traverse override

Inhibiting stop events for G331/G332

During tapping, a stop can be prevented if the block contains a path motion or a G4 as follows:

MD11550 $MN_STOP_MODE_MASK, bit 0 = 0

The stop which was activated previously is possible again after G331/G332 has been executed.

References

For further information on G331/G332, see Programming Manual Fundamentals.

16.2.5 Feedrate for tapping with compensating chuck (G63)

Function

G63 is a subfunction for tapping threads using a tap with compensating chuck. An encoder (position encoder) is not required.

Speed S, feedrate F, thread pitch

With G63, a speed S must be programmed for the spindle and a feedrate F for the infeed axis (axis for thread length).

The feedrate F must be calculated by the programmer on the basis of the speed S and the thread pitch.

Feedrate F [mm/min] = speed S [rev/min] * pitch [mm/rev]

References

For more information on G63, see Programming Manual Fundamentals.

16.3 Feedrate for positioning axes (FA)

Function

The velocity of a positioning axis is programmed with axis-specific feedrate FA.

FA is modal.

The feedrate is always G94.

Note

The maximum axis velocity (MD32000 $MA_MAX_AX_VELO) is not exceeded.

Programming

No more than five axis-specific feedrates can be programmed in each part program block.

Syntax:

```
FA[<positioning axis>] = <feedrate value>
```

<positioning axis>:	Identifier of the channel axis (MD20080 $MC_AXCONF_CHANAX_NAME_TAB)	
<feedrate value>:	Feedrate	
	Range of values:	0.001…999 999.999 mm/min, deg/min or 0.001…39 999.9999 inch/min

Default setting

If no axial feedrate FA is programmed, the axial default setting is applied:

MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)

Output to PLC

The feedrate value can be output to the the PLC:

- To the channel-specific NC/PLC interface via:

  DB21, ... DBB158 - DBB193

- To the axis-specific NC/PLC interface via:

  DB31, ... DBB78 - DBB81

The output time is specified with the machine data:

MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

The output is suppressed in the default setting (MD22240 = 3), because drops in velocity can occur through the output of F functions to the NC/PLC interface in continuous-path mode.

For more information, see Description of Functions "H2: Auxiliary function outputs to PLC (Page 395)".

Reset behavior

The behavior after the end of program or NC reset is specified by the machine data:

MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset)

Value	Meaning
0	The default values are effective after NC reset.
1	The last programmed FA values are effective after NC reset.

16.4 Feedrate control

16.4.1 Feedrate disable and feedrate/spindle stop

Function

The "Feed disable" or "Feed/spindle stop" brings the axes to a standstill with adherence to the braking characteristics and the path contour (exception: G33 block).

Channel-specific feedrate disable

The NC/PLC interface signal:
DB21, ... DBX6.0 (feedrate disable)
stops all axes (geometry and auxiliary axes) of a channel in all modes.

Effectiveness of the channel-specific feedrate disable:	
• With active G33, G34, G35:	**Not** effective
• With active G63:	Effective
• With active G331, G332:	Effective

"Feed stop" for geometry axes in JOG mode

The NC/PLC interface signals:
DB21, ... DBX12.3 (feed stop for geometry axis 1)
DB21, ... DBX16.3 (feed stop for geometry axis 2)
DB21, ... DBX20.3 (feed stop for geometry axis 3)
stop the relevant geometry axes if a channel in JOG mode.

Axis-specific "Feed stop"

The axis-specific NC/PLC interface signal:
DB31, ... DBX4.3 (feed stop)
stops the respective machine axis.

In automatic mode:

- If the "Feed stop" is performed for a path axis, all the axes traversed in the current block and all axes participating in the axis group are stopped.
- If the "Feed stop" is performed for a positioning axis, only this axis is stopped.

Only the current axis is stopped in JOG mode.

Effectiveness of the axis-specific "Feed stop":	
• With active G33, G34, G35:	Effective (causes contour deviations)
• With active G63:	Effective
• With active G331, G332:	Effective

Axis/spindle disable

With active "Axis/spindle disable":
DB31, ... DBX1.3 = 1
The axial PLC interlocks "No controller enable" and "Feed stop" have **no effect**.

However, the axial and channel-specific override are effective.

"Spindle stop"

The NC/PLC interface signal:
DB31, ... DBX4.3 (spindle stop)
stops the respective spindle.

Effectiveness of the "Spindle stop" function	
• With active G33, G34, G35:	Effective (may cause contour deviations depending on dynamic characteristics)
• With active G63:	Effective
• With active G331, G332:	**Not** effective

16.4.2 Feedrate override on machine control panel

Function

With the "Feedrate override via the machine control panel", the user can locally increase or decrease the path feedrate at the machine as a percentage with immediate effect. To achieve this, the programmed feedrates are multiplied with the override values available at the NC/PLC interface.

The feedrate can be changed axis-specifically for positioning axes.

The "Spindle override" can be used to change the spindle speed and the cutting rate (G96, G961).

With a feedrate change, the axial acceleration and velocity limits are maintained. There are no contour errors along the path.

The feedrate override can be changed separately for path and position axes.

The overrides influence the programmed values or the limits (e.g. G26, LIMS for spindle speed).

Channel-specific feedrate and rapid traverse

For feedrate and rapid traverse override, dedicated enable signals and correction/offset factors are available in the NC/PLC interface:

DB21, ... DBX6.7 (feedrate override active)

DB21, ... DBB4 (feedrate override)

DB21, ... DBX6.6 (rapid traverse override active)

DB21, ... DBB5 (rapid traverse override)

The override factors can be specified from the PLC either in the binary or gray-coded format. The format is communicated to the NC via the following machine data:

MD12020 $MN_OVR_FEED_IS_GRAY_CODE (path feedrate override switch gray-coded)

MD12040 $MN_OVR_RAPID_IS_GRAY_CODE (rapid traverse override switch gray-coded)

The following permanent assignment applies to binary code:

Binary code	decimal	Override factor
00000000	0	0.00 ≙ 0%
00000001	1	0.01 ≙ 1%
00000010	2	0.02 ≙ 2%
00000011	3	0.03 ≙ 3%
00000100	4	0.04 ≙ 4%
...	...	...
01100100	100	1.00 ≙ 100%
...	...	...
11001000	200	2.00 ≙ 200%

With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:

MD12030 $MN_OVR_FACTOR_FEEDRATE [<n>] (evaluation of the path feedrate override switch)

MD12050 $MN_OVR_FACTOR_RAPID_TRA [<n>] (evaluation of the rapid traverse override switch)

An active feedrate override has an effect on all path axes that are assigned to the current channel. An active rapid traverse override has an effect on all the axes that are traversed with rapid traverse and that are assigned to the current channel.

No rapid traverse override switch available

If there is no dedicated rapid traverse override switch, you can choose between rapid traverse override and feedrate override. The override to be active can be selected via the PLC or operator panel front. When rapid traverse override is active, the feedrate override values are limited to 100%.

- When the rapid traverse override is activated via the operator panel front, the basic PLC program:
  - Transfers the selection of the feedrate override for rapid traverse on the activation signal for the rapid traverse override:

    DB21, ... DBX6.6 = DB21, ... DBX25.3
  - Transfers the feedrate override value in the rapid traverse override value:

    DB21, ... DBB5 = DB21, ... DBB4
- When the rapid traverse override is selected via the PLC, the PLC user program:
  - Must set the activation signal for the rapid traverse override:

    DB21, ... DBX6.6 = 1
  - Must transfer the feedrate override value in the rapid traverse override value:

    DB21, ... DBB5 = DB21, ... DBB4

Effectiveness of the channel-specific feedrate and rapid traverse override:	
• With active G33, G34, G35:	**Not** effective
• With active G63:	**Not** effective
• With active G331, G332:	**Not** effective

Reference velocity for path feedrate override

The reference velocity for the "Path feedrate override via machine control panel" can be set differently to the standard feedrate (= programmed feedrate).

MD12082 $MN_OVR_REFERENCE_IS_MIN_FEED

Axis-specific feedrate override

An enable signal and a byte for the feedrate override factor are in the NC/PLC interface for each positioning axis.

DB31, ... DBX1.7 (override effective)

DB31, ... DBB0 (feedrate override)

The override factor can be specified from the PLC either in the binary or gray-coded format. The format is communicated to the NC via the following machine data:

MD12000 $MN_OVR_AX_IS_GRAY_CODE (axis feedrate override switch gray-coded)

The following permanent assignment applies to binary code:

Binary code	Decimal	Override factor
00000000	0	0.00 ≙ 0%
00000001	1	0.01 ≙ 1%
00000010	2	0.02 ≙ 2%
00000011	3	0.03 ≙ 3%
00000100	4	0.04 ≙ 4%
...	...	...
01100100	100	1.00 ≙ 100%
...	...	...
11001000	200	2.00 ≙ 200%

With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:

MD12010 $MN_OVR_ FACTOR_AX_ SPEED [<n>] (evaluation of the axis feedrate override switch)

Effectiveness of the axis-specific feedrate override:	
• With active G33, G34, G35:	**Not** effective
• With active G63:	**Not** effective (the override is set in the NC permanently to 100%)
• With active G331, G332:	**Not** effective (the override is set in the NC permanently to 100%)

Spindle override

One enable signal and one byte for the spindle override factor are available in the NC/PLC interface for each spindle.

DB31, ... DBX1.7 (override effective)

DB31, ... DBB19 (spindle override)

The override factor can be specified from the PLC either in the binary or gray-coded format. The format is communicated to the NC via the following machine data:

MD12060 $MN_OVR_SPIND_IS_GRAY_CODE (spindle override switch gray-coded)

The following permanent assignment applies to binary code:

Binary code	decimal	Override factor
00000000	0	0.00 ≙ 0%
00000001	1	0.01 ≙ 1%
00000010	2	0.02 ≙ 2%
00000011	3	0.03 ≙ 3%
00000100	4	0.04 ≙ 4%
...	...	...
01100100	100	1.00 ≙ 100%
...	...	...
11001000	200	2.00 ≙ 200%

With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:

MD12070 $MN_OVR_FACTOR_SPIND_SPEED [<n>] (evaluation of the spindle override switch)

Effectiveness of the "Spindle override":	
• With active G33, G34, G35:	Effective
• With active G63:	**Not** effective
• With active G331, G332:	Effective

Reference to spindle override

The spindle override can refer to the speed or the programmed speed limited by the machine or setting data. The setting is realized via:

MD12080 $MN_OVR_REFERENCE_IS_PROG_FEED (override reference velocity)

Limiting the override factor

For binary-coded override factors, the maximum possible overrides for path feedrate, axis feedrate and spindle speed can be limited:

MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

Override active

For overrides that have been enabled, the specified override values entered via the machine control panel become immediately active in all operating modes and machine functions.

Override inactive

An override factor of 100% is internally effective if an override is not activated. The override factor at the NC/PLC interface is not evaluated.

An exception is the zero setting for a binary interface and the 1st switch setting for a gray-coded interface. In these cases, the override factors entered at the NC/PLC interface are evaluated. For a binary interface, the override factor is always 0%. For a gray-coded interface, the value entered in machine data for the 1st switch position value is output as override value. It should be populated with 0.

16.4.3 Programmable feedrate override

Function

The "Programmable feedrate override" function can be used to change the velocity level of path and positioning axes via the part program.

Programming

Syntax	Meaning
`OVR=<value>`	Feedrate change for path feedrate F
`OVRA[<axis>=<value>`	Feedrate change for positioning feedrate FA

The programmable range is between 0 and 200%.

Default setting: 100%

Effectiveness

The NC/PLC interface signals DB21, ... DBB6 (rapid traverse or feedrate override active) and DB31, ... DBX1.7 (axis-specific override active) do **not** refer to the programmable feedrate override. The programmable feedrate override remains active when these signals are deactivated.

The effective override is calculated from the product of the "Programmable feedrate override" and the "Feedrate override on machine control panel (Page 1536)".

The default setting for the "Programmable feedrate override" is 100%.

The default setting is effective:

- If no feedrate override is programmed or
- After reset if the machine data:
  MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset)
  is not set.

Note

OVR is not effective with G33, G34, G35.

16.4.4 Dry run feedrate

Function

The dry run feedrate is used when testing part programs without machining the workpiece in order to allow the program or program sections to execute with an increased path feedrate, for example.

Activation

The dry run feedrate can be selected in the automatic modes and activated from the PLC or the operator panel front.

When activated from the operator panel front, the interface signal:
DB21, ... DBX24.6 (dry run feedrate selected)
is set and transferred from the basic PLC program to the interface signal:
DB21, ... DBX0.6 (activate dry run feedrate).

When selected on the PLC, the interface signal DB21, ... DBX0.6 (activate dry run feedrate) must be set from the PLC user program.

Effectiveness

As long as the "Activate dry run feedrate" interface signal is set, instead of the programmed feedrate, the feedrate value set via SD42100 DRY_RUN_FEED is effective in the way specified via SD42101 $SC_DRY_RUN_FEED_MODE (see parameterization):

The dry run feedrate is always interpreted as linear feedrate (G94).

Parameterization

Activation of dry run feedrate

The time of activation depends on the setting in the machine data:

MD10704 $MN_DRYRUN_MASK (activation of dry run feedrate)

Value	Meaning
0	The dry run feedrate may only be switched on and off at the end of the block (default setting).
1	The dry run feedrate can also be activated during the program processing (in the part program block). **Notice:** Activation during processing triggers an internal reorganization operation on the controller which causes the axes to be stopped for a short time. This can affect the surface finish of the workpiece being machined.
2	The dry run feedrate can be activated/deactivated at any time without the axes being stopped. The function only takes effect with a block "later" in the program run.

Changing the dry run feedrate

The feedrate for the dry run is entered in the setting data:

SD42100 $SC_DRY_RUN_FEED (dry run feedrate)

The setting data can be changed via the operator panel front in the "Parameters" operating area.

If the selection has been accepted by the NCK, the following NC/PLC interface signal is set:

DB21, ... DBX318.6 (dry run feedrate active)

"DRY" is displayed in the operator panel front status bar to indicate an active dry run feedrate if:

- Selection took place during the program stop at the end of a block or
- The machine data MD10704 $MN_DRYRUN_MASK was set to "1" during the program execution

Mode of operation of the dry run feedrate

The mode of operation of the dry run feedrate entered in SD42100 can be set via the setting data:

SD42101 $SC_DRY_RUN_FEED_MODE

Value	Meaning
0	The programmed feedrate is compared to the dry run feedrate in SD42100 and then traversing is performed with the higher of the two feedrates (default setting).
1	The programmed feedrate is compared to the dry run feedrate in SD42100 and then traversing is performed with the lower of the two feedrates.
2	The dry run feedrate entered in SD42100 takes effect directly, irrespective of the programmed velocity.
3 -9	Reserved
10	As for configuration 0, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.
11	As for configuration 1, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.
12	As for configuration 2, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.

16.4.5 Multiple feedrate values in one block

Function

The function "Multiple feedrate values in one block" can be used to activate six different feedrate values of an NC block, a dwell time or a retraction motion-synchronously, depending on the external digital and/or analog inputs.

When the input for the sparking out time or retraction path is activated, the distance-to-go for the path axes or the particular single axis is deleted and the dwell time or retraction is started.

The retraction is started within an IPO cycle.

Signals

The input signals are combined in one input byte for the function. A fixed functional assignment applies within the byte.

Table 16- 1 Input byte for the "Multiple feedrates in one block" function

	Bit							
	7	6	5	4	3	2	1	0
Input no.	I7	I6	I5	I4	I3	I2	I1	I0
Feedrate address	F7	F6	F5	F4	F3	F2	ST	SR

I7 to I2: Activation of feedrates F7 to F2

E1: Activation of the dwell time ST/STA (in seconds)

I0: Activation of the retraction motion SR/SRA

Priority of the signals

The signals are scanned in ascending order starting at I0. Therefore, the retraction motion (SR) has the highest priority and the feedrate F7 the lowest priority.

SR and ST end the feedrate motions that were activated with F2 to F7.
SR also ends ST, i.e. the complete function.

The signal with the highest priority determines the current feedrate.

The response to loss of the respective highest-priority input (F2 - F7) can be defined with the machine data:

MD21230 $MC_MULTFEED_STORE_MASK (storage behavior for the "Multiple feedrate values in one block" function)

Bit	Value	Meaning
2 ... 7	0	With the loss of the respective highest-priority input, the associated feedrate is not retained (default setting).
	1	Set bit 2 to 7 ensures that the associated feedrate (F2 to F7) that was selected by the respective highest-priority input signal is also retained when there is a loss of the input signal and a lower-priority input is active.

The end-of-block criterion is satisfied when:

- The programmed end position is reached
- The retraction motion ends (SR)
- The dwell time elapses (ST)

Hardware assignment

The input byte for the "Multiple feedrate values in one block" function can be assigned a maximum of two digital input bytes or comparator input bytes of the NCK I/O:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN (assignment of the input bytes of the NCK I/O for "Multiple feedrate values in one block"), bit 0 ... 15

The input bits can also be inverted:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN, bit 16 ... 31

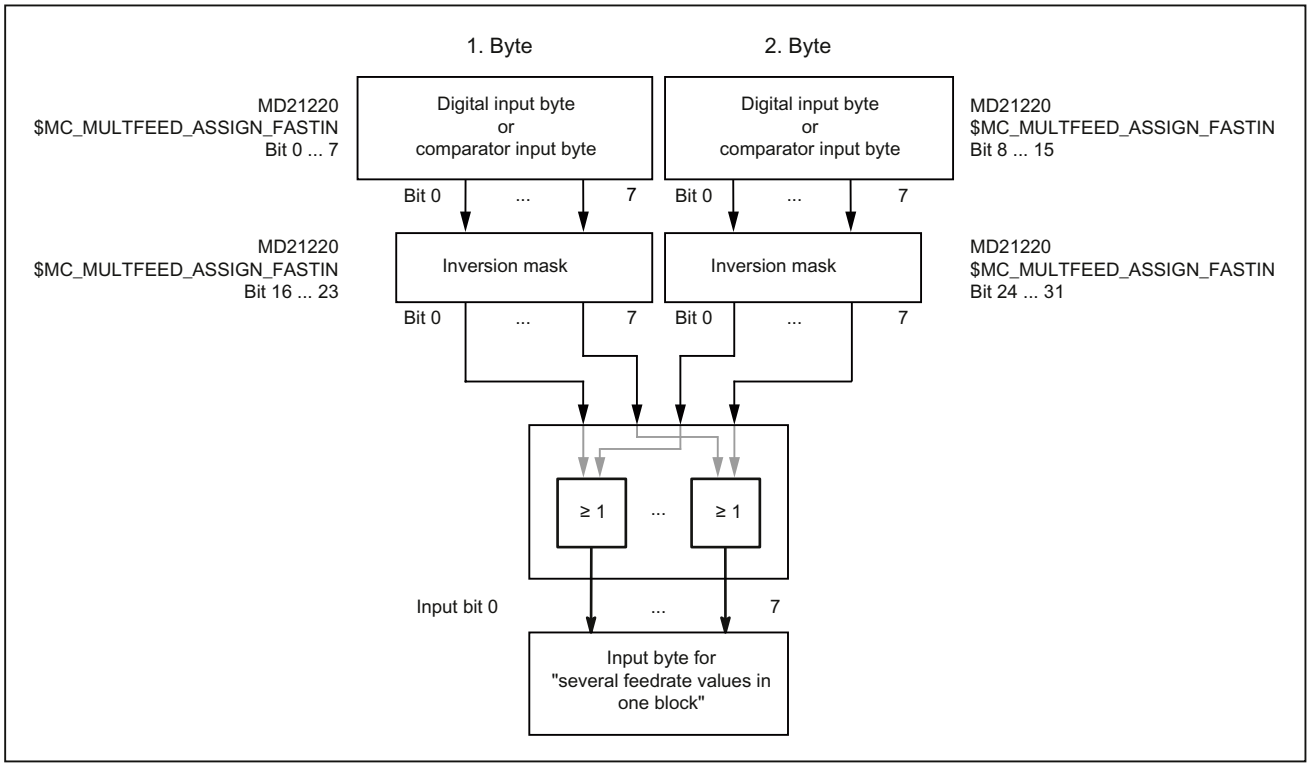


Figure 16-2 Signal assignment for the "Multiple feedrate values in one block" function

The assignment of the digital input bytes and parameterization of the comparators are described in:

References:

Function Manual, Extended Functions; Digital and Analog NCK I/O (A4)

Programming

Path motion

The path feedrate is programmed under the address F and remains valid until an input signal is present. This value acts modally.

F2=... to F7=... can be used in addition to the path feedrate to program up to six further feedrates in the block. The numerical expansion indicates the bit number of the input that activates the feedrate when changed:

Example:

```
F7=1000          ;7 corresponds to input bit 7
```

The programmed values act non-modally. The path feedrate programmed under F applies in the next block.

Dwell (sparking out time) and retraction path are programmed under separate addresses in the block:

ST=...	Dwell time (for grinding sparking out time)
SR=...	Retraction path

These addresses apply non-modally.

Axial motion

The axial feedrates are programmed under address FA and remain valid until an input signal is present. They act modally.

FMA[2,<axis>]=... to FMA[7,<axis>]=... can be used to program up to six further feedrates per axis in the block.

The first expression in square brackets indicates the bit number of the input that activates the feedrate when changed. The second expression indicates the axis to which the feedrate applies.

Example:

```
FMA[3,Y]=1000        ; Axial feedrate for Y axis, corresponds to input bit 3
```

The values programmed under FMA act non-modally. The feedrate programmed under FA applies to the next block.

Dwell (sparking out time) and retraction path can also be defined for a single axis:

`STA[<axis>]=...`	Axial dwell time (sparking out time)
`SRA[<axis>]=...`	Axial retraction path

The expression in square brackets indicates the axis for which the sparking out time and retraction path apply.

Examples:

```
STA[X]=2.5      ; The sparking out time for the X axis is 2.5 seconds.
SRA[X]=3.5      ; The retraction path for the X axis is 3.5 (unit e.g. mm).
```

These addresses apply non-modally.

Note

Retraction path

The unit for the retraction path refers to the current valid unit of measurement (mm or inch).

The reverse stroke is always made in the opposite direction to the current motion. `SR`/`SRA` always programs the value for the reverse stroke. No sign is programmed.

Note

POS instead of POSA

If feedrates, sparking out time (dwell time) or return path are programmed for an axis on account of an external input, this axis must not be programmed as POSA axis (positioning axis over multiple blocks) in this block.

Note

Status query

It is also possible to poll the status of an input for synchronous commands of various axes.

Note

LookAhead

Look Ahead is also active for multiple feedrates in one block. In this way, the current feedrate can be restricted by the Look Ahead value.

Application

The "Multiple feedrate values in one block" function is used primarily for grinding, but is not restricted to it.

Typical applications are, for example:

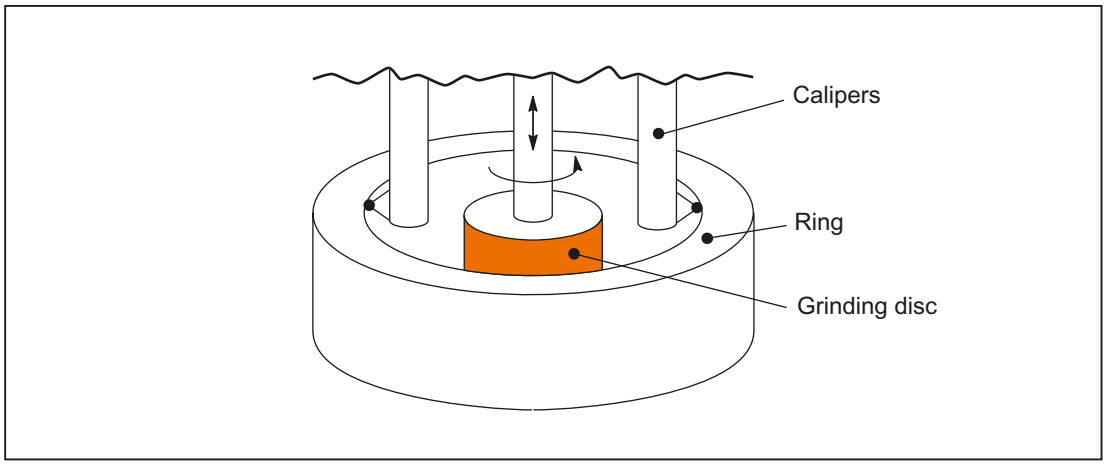
- Analog or digital calipers

  Depending on whether the external inputs are analog or digital, various feedrate values, a dwell time and a retraction path can be activated. The limit values are defined via the setting data.

- Switching from infeed to working feedrate via proximity switch

Example

Internal grinding of a conical ring, where the actual diameter is determined using calipers and, depending on the limits, the feedrate value required for roughing, finishing or fine finishing is activated. The position of the calipers also provides the end position. Thus, the block end criterion is determined not only by the programmed axis position of the infeed axis but also by the calipers.




```
G00 X40 F=1000                                          ; Initial setting
G01 X48 F=20 F7=5 F3=2.5 F2=0.5 ST=1.5 SR=0.5
                                                  Retraction
                                              Sparking out time
                                        Feedrate fine finishing
                                   Feedrate finishing
                              Feedrate roughing
                         Feedrate air grinding
```

16.4.6 Fixed feedrate values

Function

The "Fixed feedrate values" function can be used to activate fixed feedrates (max. four) defined via the machine data instead of the programmed feedrate or the configured JOG velocities.

The function is available in AUTOMATIC and JOG mode.

Behavior in AUTOMATIC mode

The contour travels at the activated fixed feedrate, instead of using the programmed feedrate.

Behavior in JOG mode

The axis is traversed with the activated fixed feedrate instead of the configured JOG velocity / JOG rapid traverse velocity. The travel direction is specified via the interface signal.

Parameterization

The setting of the fixed feedrates is performed:

- For linear axes with the machine data:

  MD12202 $MN_PERMANENT_FEED[<n>]

- For rotary axes with the machine data:

  MD12204 $MN_PERMANENT_ROT_AX_FEED[<n>]

where <n> = 0, 1, 2, 3 (for fixed feedrate 1, 2, 3, 4)

Note

The fixed feedrates are always linear feedrate values. Switchover to linear feedrate is conducted internally even in case of revolutional feedrate.

Activation

The fixed feedrates are activated via NC/PLC interface signals:

- In AUTOMATIC mode for path/geometry axes using the channel-specific interface signals:

  DB21, ... DBX29.0 (activate fixed feedrate 1)

  DB21, ... DBX29.1 (activate fixed feedrate 2)

  DB21, ... DBX29.2 (activate fixed feedrate 3)

  DB21, ... DBX29.3 (activate fixed feedrate 4)

- In JOG mode for machine axes using the axis-specific interface signals:

  DB31, ... DBX3.2 (activate fixed feedrate 1)

  DB31, ... DBX3.3 (activate fixed feedrate 2)

  DB31, ... DBX3.4 (activate fixed feedrate 3)

  DB31, ... DBX3.5 (activate fixed feedrate 4)

Supplementary conditions

Effectiveness

The function "Fixed feedrate values" is **not** active:

- For spindles
- For positioning axes
- When tapping

Override = 0

The traversing behavior for override = 0 depends on the setting in machine data:

MD12200 $MN_RUN_OVERRIDE_0

DRF offset

The DRF offset cannot be activated for a selected fixed feedrate.

16.4.7 Programmable feedrate characteristics

Function

To permit flexible definition of the feedrate characteristic, the feedrate programming according to DIN 66025 has been extended by linear and cubic characteristics.

The cubic profiles can be programmed directly or as an interpolating spline.

Programming

You can program the following feedrate profiles:

- FNORM

  Behavior in accordance with DIN 66025 (default setting).

  An F-value programmed in the block is applied over the entire path of the block, and is subsequently regarded as a fixed modal value.

- FLIN

  An F value programmed in the block is traversed linearly over the path from the current value at the beginning of the block to the end of the block, and is subsequently regarded as modal value.

- FCUB

  The non-modal programmed F values (relative to the end of the block) are connected by a spline. The spline starts and ends tangentially to the previous or following feedrate setting. If the F address is missing in one block, then the last programmed F value is used.

- FPO

  The F address [syntax: `F=FPO(...,...,...)`] designates the characteristic of the feedrate via a polynomial from the current value to the end of the block in which it was programmed. The end value is treated as modal from there onwards.

Parameterization

If FLIN and FCUB are used in connection with compression COMPON, a tolerance can be defined for the path feedrate:

MD20172 $MC_COMPRESS_VELO_TOL (max. permissible deviation of the path feedrate with compression)

Supplementary conditions

FLIN/FCUB

The path velocity profile programmed with FLIN or FCUB is not active together with revolutional feedrate for G95 as well as with constant cutting rate with G96/G961 and G97/G971.

References

For further information on the programmable feedrate characteristics, see Programming Manual, Job Planning.

16.4.8 Feedrate for chamfer/rounding FRC, FRCM

The machining conditions can change significantly during surface transitions to chamfer/rounding. Hence, the chamfer/rounding contour elements require dedicated, optimized feedrate values to achieve the desired surface quality.

Function

The feedrate for chamfer/rounding can be programmed via NC addresses.

Programming

Syntax:

```
... FRC/FRCM=<value>
```

Meaning:

`FRC`:	**Non-modal** feedrate for chamfer/rounding
`FRCM`:	**Modal** feedrate for chamfer/rounding
`<value>`:	The feedrate is interpreted according to the active feedrate type:
	• G94, G961, G971: Feedrate in mm/min or inch/min or °/min
	• G95, G96, G97: Revolutional feedrate in mm/rev or inch/rev

Note

FRC is only effective if a chamfer/rounding is programmed in the block or if RNDM has been activated.

FRC overwrites the F or FRCM value in the current block.

The feedrate programmed under FRC must be greater than zero.

FRCM=0 activates the feedrate programmed under F for chamfering/rounding.

Parameterization

Assignment of the chamfer/rounding to the previous or following block

The feedrate type (G94, G95, G96, G961 ...) and therefore the conversion to the internal format must be consistent within the block for F and FRC/FRCM. In this context, the following machine data must be taken into account:

MD20201 $MC_CHFRND_MODE_MASK (chamfer/rounding behavior)

Bit	Value	Meaning
0	0	The technology of the chamfer/rounding (feedrate, feedrate type, M commands, etc.) is determined by the following block (default setting).
	1	The technology of the chamfer/rounding is determined by the previous block (recommended setting).

Maximum number of empty blocks

The number of blocks without traversing information in the compensation plane (empty blocks) permitted between two blocks with traversing information during active chamfer/rounding, is limited. The maximum number is specified in the machine data:

MD20200 $MC_CHFRND_MAXNUM_DUMMY_BLOCKS (empty blocks for chamfer/radii)

Supplementary conditions

FLIN/FCUB

Feedrate interpolation FLIN and FCUB is **not** possible for chamfer/rounding.

G0

FRC/FRCM is not active when a chamfer is traversed with G0. The programming is possible in accordance with the F value without error message.

Change G94 ↔ G95

If FRCM is programmed, the FRCM value will need to be reprogrammed like F on change G94 ↔ G95, etc. If only F is reprogrammed and if the feedrate type FRCM > 0 before the change, an error message will be output.

Example

Example 1: MD20201 bit 0 = 0; take feedrate from following block (default setting!)

```
Program code                        Comment
N10 G0 X0 Y0 G17 F100 G94
N20 G1 X10 CHF=2                    ; Chamfer N20-N30 with F=100 mm/min
N30 Y10 CHF=4                       ; Chamfer N30-N40 with FRC=200 mm/min
N40 X20 CHF=3 FRC=200               ; Chamfer N40-N60 with FRCM=50 mm/min
N50 RNDM=2 FRCM=50
N60 Y20                             ; Modal rounding N60-N70 with FRCM=50 mm/min
N70 X30                             ; Modal rounding N70-N80 with FRCM=50 mm/min
N80 Y30 CHF=3 FRC=100               ; Chamfer N80-N90 with FRC=100 mm/min
N90 X40                             ; Modal rounding N90-N100 with F=100 mm/min
                                      (deselection of FRCM)
N100 Y40 FRCM=0                     ; Modal rounding N100-N120 with G95 FRC=1 mm/rev
N110 S1000 M3
N120 X50 G95 F3 FRC=1
...
M02
```

Example 2: MD20201 bit 0 = 1; take feedrate from previous block (recommended setting!)

```
Program code                        Comment
N10 G0 X0 Y0 G17 F100 G94
N20 G1 X10 CHF=2                    ; Chamfer N20-N30 with F=100 mm/min
N30 Y10 CHF=4 FRC=120               ; Chamfer N30-N40 with FRC=120 mm/min
N40 X20 CHF=3 FRC=200               ; Chamfer N40-N60 with FRC=200 mm/min
N50 RNDM=2 FRCM=50
N60 Y20                             ; Modal rounding N60-N70 with FRCM=50 mm/min
N70 X30                             ; Modal rounding N70-N80 with FRCM=50 mm/min
N80 Y30 CHF=3 FRC=100               ; Chamfer N80-N90 with FRC=100 mm/min
N90 X40                             ; Modal rounding N90-N100 with FRCM=50 mm/min
N100 Y40 FRCM=0                     ; Modal rounding N100-N120 with F=100 mm/min
N110 S1000 M3
N120 X50 CHF=4 G95 F3 FRC=1         ; Chamfer N120-N130 with G95 FRC=1 mm/rev
N130 Y50                            ; Modal rounding N130-N140 with F=3 mm/rev
N140 X60
...
M02
```

16.4.9 Non-modal feedrate FB

Function

The "Non-modal feedrate" function can be used to define a separate feedrate for a single part program block. After this block, the previous modal path feedrate is active again.

Programming

Syntax:

```
... FB=<value>
```

Meaning:

FB:	Separate feedrate for the current block
<value>:	The feedrate is interpreted according to the active feedrate type:
	• G94, G961, G971: Feedrate in mm/min or inch/min or °/min
	• G95, G96, G97: Revolutional feedrate in mm/rev or inch/rev

Note

The feedrate programmed under FB must be greater than zero.

If no traversing motion is programmed in the block (e.g. computation block), the FB has no effect.

If no explicit feed for chamfering/rounding is programmed, then the value of FB also applies for any contour element chamfering/rounding in this block.

Simultaneous programming of FB and FD (handwheel travel with feedrate override) or F (modal path feedrate) is not possible.

16.4.10 Influencing the single axis dynamic response

Single axes

Single axes can be programmed in the part program, in synchronized actions and via the PLC:

- Part program:

```
POS[<axis>]=...
POSA[<axis>]=...
SPOS[<axis>]=...
SPOSA[<axis>]=...
OS[<axis>]=...
OSCILL[<axis>]=...
```

- Synchronized actions:

```
EVERY ... DO
POS[<axis>]=...
SPOS[<spindle>]=...
MOV[<axis>]=...
```

- PLC: FC18

Dynamic response

The dynamic response of an axis is influenced by:

- MD32060 $MA_POS_AX_VELO (positioning axis velocity)

  The effective positioning axis velocity can be changed:

  – Part program / synchronized action: Axial feedrate FA or percentage feedrate override OVRA

  – PLC: Specification of FRate or overwriting the axial override

- MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)

  The effective maximum axis acceleration can be changed:

  – Part program indirectly: Writing the machine data with subsequent NewConfig

  – Part program directly: Percentage acceleration override ACC

  – Synchronized actions indirectly: Writing the machine data and initiating an ASUB for the activation of NewConfig

  – Synchronized actions directly: Percentage acceleration override `ACC` (cannot be preset by the PLC).

  Via the PLC, the same options apply as in synchronized actions.

- Part program commands: BRISKA, SOFTA, DRIVEA, JERKA

  Cannot be programmed in synchronized actions (only indirectly via ASUB).

  Cannot be specified by the PLC (only indirectly via ASUB).

- Active servo parameter set

  The active parameter set can be changed:

  – Part program / synchronized action: SCPARA

  – PLC: DB31, … DBX9.0-2 (controller parameter set)

  For detailed information on the servo parameter sets, see "Parameter sets of the position controller (Page 381)".

Note

Dynamic response changes

Dynamic response changes made in the part program do not affect command or PLC axis motion. Dynamic response changes made in synchronized actions have no effect on traversing motion programmed in the part program.

Feedforward control

The type of feedforward control and the path axes that should be traversed with feedforward control can be directly programmed in the part program using FFWON/FFWOF. In synchronized actions and from the PLC, programming is only possible indirectly via an ASUB.

Percentage acceleration override (ACC)

In a part program or synchronized action, the acceleration specified in machine data:
MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)
can be changed in a range from 0% – 200% using the ACC command.

Syntax:

```
ACC[<axis>]=<value>
```

Meaning:

ACC:	Keyword for the programming of the percentage acceleration override
<axis>:	Name of the channel axis or spindle
<value>:	Acceleration change in percent relative to MD32300 Range of values: 0 ... 200

The actual axial acceleration value can be read via the system variable $AA_ACC. It is determined by:

$AA_ACC[<axis>] = (MD32300 $MA_MAX_AX_ACCEL[<axis>]) * ACC[<axis>] / 100

MD32320 $MA_DYN_LIMIT_RESET_MASK can be used to specify the initial setting of the value programmed with ACC for a channel reset or end of part program M30.

Note

The acceleration override programmed with ACC can be read using the system variable $AA_ACC. However, $AA_ACC is read in the part program at a different time than when reading in a synchronized action.

The system variables $AA_ACC only contain the value programmed in the part program with ACC if, in the meantime, the acceleration override was not changed by programming ACC in a synchronized action. The same applies for the reverse situation.

Percentage acceleration override and main run axes

Depending on whether the system variable $AA_ACC is read in the part program or synchronized action, the value for the acceleration override programmed with ACC is output for the NC axes or main run axes (command axes, PLC axes, asynchronous oscillating axes, etc.).

For correct results, system variable $AA_ACC must therefore always be read at the same location (part program or synchronized action) from where the acceleration override was programmed with ACC.

Examples:

Writing ACC in a part program:

```
N80 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]
```

Writing ACC in a synchronized action:

```
N100 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]
```

Writing ACC and reading $AA_ACC in a part program:

```
ACC[X]=50                                          ; writing
RO=$AA_ACC[X]                                      ; reading

IF (RO <> $MA_MAX_AX_ACCEL[X] * 0.5)               ; checking
 SETAL(61000)
ENDIF
```

```
Writing ACC and reading $AA_ACC in a synchronized action:
WHEN TRUE DO ACC[X]=25 R0=$AA_ACC[X]                ; writing and reading
G4 F1

IF (RO <> $MA_ MAX_AX_ACCEL[X] * 0.25)              ; checking
 SETAL(61001)
ENDIF
```

end-of-motion criterion for single axes

Similar to the block change criterion for path interpolation (G601, G602, G603) the end-of-motion criterion for traversing motion of individual axes can be programmed in part programs / synchronized actions:

Program command	End-of-motion criterion
FINEA[<axis>]	"Exact stop fine"
COARSEA[<axis>]	"Exact stop coarse"
IPOENDA[<axis>]	"Interpolator stop" (IPO stop)

The most recently programmed value is kept after the end of program or NC reset.

The effective end-of-motion criterion can be read using the axis-specific system variable $AA_MOTEND.

Note

Depending on whether the system variable $AA_MOTEND is read in the part program or synchronized action, it contains the value for the NC axes or the main run axes.

Example:

```
Part program:
N80 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 COARSEA[X]
```

```
Synchronized action:
N100 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]
```

References:

For further information on block changes and end-of-motion criteria for FINEA, COARSEA and IPOENDA, see:
Function Manual, Extended Functions; Positioning Axes (P2), Section: Block change

Programmable servo parameter set (SCPARA)

In the part program / synchronized action, the servo parameter set can be specified using SCPARA.

Syntax

```
SCPARA[<axis>] = <parameter set number>
```

Meaning

SCPARA:	Keyword for the activation of the specified servo parameter set
<axis>:	Name of the channel axis
<parameter set number>:	Number of the servo parameter set to be activated

Note

The activation of the parameter set specified using SCPARA can be suppressed from the PLC user program:

DB31,… DBX9.3 = 1 (parameter set specification disabled by SCPARA)

In this case, **no** message is displayed.

The number of the active parameter set can be read using the system variable $AA_SCPAR.

Supplementary conditions

Different end-of-motion criteria

Different end-of-motion criteria will affect how quickly or slowly part program blocks are completed. This can have side effects for technology cycles and PLC user parts.

Parameter set change

The PLC user program must be expanded if the servo parameter set is to be changed both inside a part program or synchronized action and the PLC.

Power On

After POWER ON, the following initial settings are made:

- Percentage acceleration override for all single-axis interpolations: 100%
- End-of-motion criterion for all single-axis interpolations: FINEA
- Servo parameter set: 1

Mode change

When the operating mode is changed from AUTOMATIC to JOG, the programmed dynamic response changes remain valid.

Reset

With reset, the last programmed value remains for the part program specifications. The settings for main-run interpolations do not change.

Block search

The last end-of-motion criterion programmed for an axis is collected and output in an action block. The last block with a programmed end-of-motion criterion that was processed in the search run serves as a container for all programmed end-of-motion criteria for all axes.

16.5 Supplementary conditions

Unit of measurement

The valid unit of measurement of the feedrates depends on the set measuring system and the entered axis type:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC (basic system of the control metric/inch)

MD30300 $MA_IS_ROT_AX (rotary or linear axis)

Initial setting for the feedrate type

The initial setting for the feedrate type is specified in the machine data:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

The default setting is G94.

The initial setting of the feedrate type is only displayed when a part program is started.

Effectiveness after reset

Whether the last programmed F, FA, OVR, OVRA values are also active after reset depends on the setting in the machine data:

MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset)

Spindle positioning

With active G95, G96, G961, G97, G971, G33, G34, G35 spindle positioning should not be performed, because the derived path feedrate following spindle positioning = 0. If the programmed axis position has not then been reached, the block cannot be completed.

16.6 Data lists

16.6.1 Machine data

16.6.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10704	DRYRUN_MASK	Activation of dry run feedrate
10710	PROG_SD_RESET_SAVE_TAB	Setting data to be updated
11410	SUPPRESS_ALARM_MASK	Mask for suppressing special alarms
11550	STOP_MODE_MASK	Defines the stop behavior
12000	OVR_AX_IS_GRAY_CODE	Axis feedrate override switch, graycoded
12010	OVR_FACTOR_AX_SPEED	Evaluation of the axis feed override switch
12020	OVR_FEED_IS_GRAY_CODE	Path feed override switch, graycoded
12030	OVR_FACTOR_FEEDRATE	Evaluation of the path feed override switch
12040	OVR_RAPID_IS_GRAY_CODE	Rapid traverse override switch, graycoded
12050	OVR_FACTOR_RAPID_TRA	Evaluation of the rapid traverse override switch
12060	OVR_SPIND_IS_GRAY_CODE	Spindle override switch, graycoded
12070	OVR_FACTOR_SPIND_SPEED	Evaluation of the spindle override switch
12080	OVR_REFERENCE_IS_PROG_FEED	Override reference velocity
12082	OVR_REFERENCE_IS_MIN_FEED	Defining the reference of the path override
12090	OVR_FUNCTION_MASK	Selection of override specifications
12100	OVR_FACTOR_LIMIT_BIN	Limit for binarycoded override switch
12200	RUN_OVERRIDE_0	Traversing with override 0
12202	PERMANENT_FEED	Fixed feedrates for linear axes
12204	PERMANENT_ROT_AX_FEED	Fixed feedrates for rotary axes

16.6.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20100	DIAMETER_AX_DEF	Geometry axes with transverse axis functions
20150	GCODE_RESET_VALUES	Initial setting of the G groups
20172	COMPRESS_VELO_TOL	Maximum permissible deviation from path feed for compression
20200	CHFRND_MAXNUM_DUMMY_BLOCKS	Empty blocks with phase/radii
20201	CHFRND_MODE_MASK	Behavior for chamfer/rounding
20750	ALLOW_GO_IN_G96	G0 logic for G96, G961
21200	LIFTFAST_DIST	Traversing path for fast retraction from the contour
21220	MULTFEED_ASSIGN_FASTIN	Assignment of input bytes of NCK I/O for "Multiple feedrate values in one block"
21230	MULTFEED_STORE_MASK	Storage behavior for the "Multiple feedrate values in one block" function
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions
22410	F_VALUES_ACTIVE_AFTER_RESET	F function active after reset

16.6.1.3 Axis/Spindle-specific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis/spindle
32000	MAX_AX_VELO	Maximum axis velocity
32060	POS_AX_VELO	Initial setting for positioning axis velocity
32300	MAX_AX_ACCEL	Axis acceleration
32320	DYN_LIMIT_RESET_MASK	Reset behavior of dynamic limits
34990	ENC_ACTIVAL_SMOOTH_TIME	Smoothing time constant for actual values
35100	SPIND_VELO_LIMIT	Maximum spindle speed
35130	GEAR_STEP_MAX_VELO_LIMIT	Maximum speed of gear stage
35140	GEAR_STEP_MIN_VELO_LIMIT	Minimum speed of gear stage
35160	SPIND_EXTERN_VELO_LIMIT	Spindle-speed limitation via PLC

16.6.2 Setting data

16.6.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42000	THREAD_START_ANGLE	Start angle for thread
42010	THREAD_RAMP_DISP	Acceleration behavior of axis when thread cutting
42100	DRY_RUN_FEED	Dry run feedrate
42101	DRY_RUN_FEED_MODE	Dry run feed mode
42110	DEFAULT_FEED	Default value for path feed
42600	JOG_FEED_PER_REV_SOURCE	Revolutional feedrate control in the JOG mode
43300	ASSIGN_FEED_PER_RES_SOURCE	Revolutional feedrate for positioning axes/spindles

16.6.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43210	SPIND_MIN_VELO_G25	Programmed spindle speed limiting G25
43220	SPIND_MAX_VELO_G26	Programmed spindle speed limiting G26
43230	SPIND_MAX_VELO_LIMS	Spindle speed limiting with G96

16.6.3 Signals

16.6.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate dry run feedrate	DB21, ...DBX0.6	DB3200.DBX0.6
Feedrate override	DB21, ...DBB4	DB3200.DBB4
Rapid traverse override	DB21, ...DBB5	DB3200.DBB5
Feed disable	DB21, ...DBX6.0	DB3200.DBX6.0
Rapid traverse override active	DB21, ...DBX6.6	DB3200.DBX6.6
Feedrate override active	DB21, ...DBX6.7	DB3200.DBX6.7
Feed stop, geometry axis 1	DB21, ...DBX12.3	DB3200.DBX1000.3
Feed stop, geometry axis 2	DB21, ...DBX16.3	DB3200.DBX1004.3
Feed stop, geometry axis 3	DB21, ...DBX20.3	DB3200.DBX1008.3

16.6.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Dry run feedrate selected	DB21, ...DBX24.6	DB1700.DBX0.6
Feedrate override for rapid traverse selected	DB21, ...DBX25.3	DB1700.DBX1.3
Activate fixed feedrate 1 for path/geometry axes	DB21, ...DBX29.0	-
Activate fixed feedrate 2 for path/geometry axes	DB21, ...DBX29.1	-
Activate fixed feedrate 3 for path/geometry axes	DB21, ...DBX29.2	-
Activate fixed feedrate 4 for path/geometry axes	DB21, ...DBX29.3	-
Dry run feedrate active	DB21, ...DBX318.6	DB3300.DBX4002.6

16.6.3.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feed/spindle override	DB31, ...DBB0	DB380x.DBB0
Override active	DB31, ...DBX1.7	DB380x.DBX1.7
Activate fixed feedrate 1 for machine axis	DB31, ...DBX3.2	-
Activate fixed feedrate 2 for machine axis	DB31, ...DBX3.3	-
Activate fixed feedrate 3 for machine axis	DB31, ...DBX3.4	-
Activate fixed feedrate 4 for machine axis	DB31, ...DBX3.5	-
Feed stop/spindle stop	DB31, ...DBX4.3	DB380x.DBX4.3

16.6.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Revolutional feedrate active	DB31, ...DBX62.2	DB390x.DBX2.2
F function for positioning axis	DB31, ...DBB81	-
Programmed speed too high	DB31, ...DBX83.1	DB390x.DBX2001.1

W1: Tool offset

17

17.1 Brief description

Calculating tool compensation data

The SINUMERIK 840D sl controller can be used to calculate the following tool compensation data:

- Length compensation
- Radius compensation
- Storage of tool data in a flexible tool offset memory:
  - Tool identification with T numbers from 0 to 32000
  - Definition of a tool with a maximum of 9 cutting edges
  - Cutting edge described by up to 25 tool parameters

- Tool selection selectable: Immediate or via selectable M function
- Tool radius compensation:
  - Selection and deselection strategy configurable: Normal or contour-related
  - Compensation active for all interpolation types:

    Linear

    Circle

    Helical

    Spline

    Polynomial

    Involute
  - Compensation at outer corners selectable:

    Transition circle/ellipse (G450) or equidistant intersection (G451)
  - Parameter-driven adaptation of G450/G451 functions to the contour
  - Free traversing on outer corners with G450 and DISC parameter
  - Number of dummy blocks without axis motion selectable in the compensation plane
  - Collision detection selectable:

    Possible contour violations are detected predictively, if:

    - Path is shorter than tool radius

    - Width of an inside corner is shorter than the tool diameter
  - Keep tool radius compensation constant
  - Intersection procedure for polynomials

Toolholder with orientation capability

This function permits the machining of inclined surfaces with allowance for tool length compensation, provided that the kinematics of the toolholder (without NC axes) permits a static orientation of the tool. The more complex 5-axis transformation is not required for this case.

Reference:
Function Manual, Special Functions; Multi-Axis Transformations (F2)

Appropriate selection of the tool data and toolholder data describes the kinematics for the controller such that it can make allowance for the tool length compensation. The controller can take some of the description data direct from the current frame.

Note

Please refer to the following documentation for further information on tools and tool compensations and a full technical description of the general and specific programming features for tool compensation (TLC and TRC):

References:
Programming Manual, Fundamentals

Flat/unique D number structure

Compensations can be selected via unique D numbers with management function.

Special handling of tool compensations

The evaluation of signs can be controlled for tool length and wear by the setting data:

SD42900 $SC_MIRROR_TOOL_LENGTH (sign change tool length when mirroring)

SD42960 $SC_TOOL_TEMP_COMP (temperature compens. regarding tool).

The same applies to the response of the wear components when mirroring geometry axes or changing the machining plane via setting data.

References:
Programming Manual, Fundamentals, Tool Offsets

G461/G462

In order to enable the solid machining of inside corners in certain situations with the activation and deactivation of tool radius compensation, commands `G461` und `G462` have been introduced and the approach/retraction strategy has thus been extended for tool radius compensation.

- `G461`

  If no intersection is possible between the last TRC block and a previous block, the controller calculates an intersection by extending the offset curve of this block with a circle whose center point coincides with the end point of the noncorrected block, and whose radius is equal to the tool radius.

- `G462`

  If no intersection is possible between the last TRC block and a previous block, the controller calculates an intersection by inserting a straight line at the end point of the last block with tool radius compensation (the block is extended by its end tangent).

Changing from G40 to G41/42

The change from G40 to G41/G42 and vice versa is no longer treated as a tool change for tools with relevant tool point direction (turning and grinding tools).

Tool compensation environments

Functions which enable the following actions in relation to the current states of tool data are available in SW 7.1:

- Save
- Delete
- Read
- Modify

Some of the functions were previously implemented in measuring cycles. They are now universally available.

A further function can be used to determine information about the assignment of the tool lengths of the active tool to the abscissa, ordinate and applicate.

17.2 Tool

17.2.1 General

Select tool

A tool is selected in the program with the T function.

Whether the new tool will be loaded immediately by means of the T function depends on the setting in the machine data:

$MC_TOOL_CHANGE_MODE (new tool compensation with M function) determines whether the new tool is loaded immediately on execution of the T function.

Change tool immediately

MD22550 $MC_TOOL_CHANGE_MODE = 0 (new tool compensation with M function).

The new tool is changed immediately with the T function.

This setting is used mainly for turning machines with tool revolver.

Change tool with M06

MD22550 $MC_TOOL_CHANGE_MODE = 1 (new tool compensation with M function).

The new tool is prepared for changing with the T function.

This setting is used mainly on milling machines with a tool magazine, in order to bring the new tool into the tool change position without interrupting the machining process.

The old tool is removed from the spindle and the new tool is loaded into the spindle with the entered M function in the machine data:

MD22560 $MC_TOOL_CHANGE_M_CODE (M function for tool change)

This tool change must be programmed with the M function `M06`, in accordance with DIN 66025.

The next tool is preselected with the machine data:

MD20121 $MC_TOOL_PRESEL_RESET_VALUE (Preselected tool at RESET)

Its tool length compensation values must be considered at `RESET` and powerup according to machine data:

MD20110 $MC_RESET_MODE_MASK (Determination of control default settings after RESET/TP end).

Value range of T

The T function accepts the following whole numbers:

- From `T0` (no tool)
- To `T32000` (tool number 32000).

Tool cutting edge

Each tool can have up to 9 cutting edges. The 9 tool cutting edges are assigned to the D functions D1 to D9.

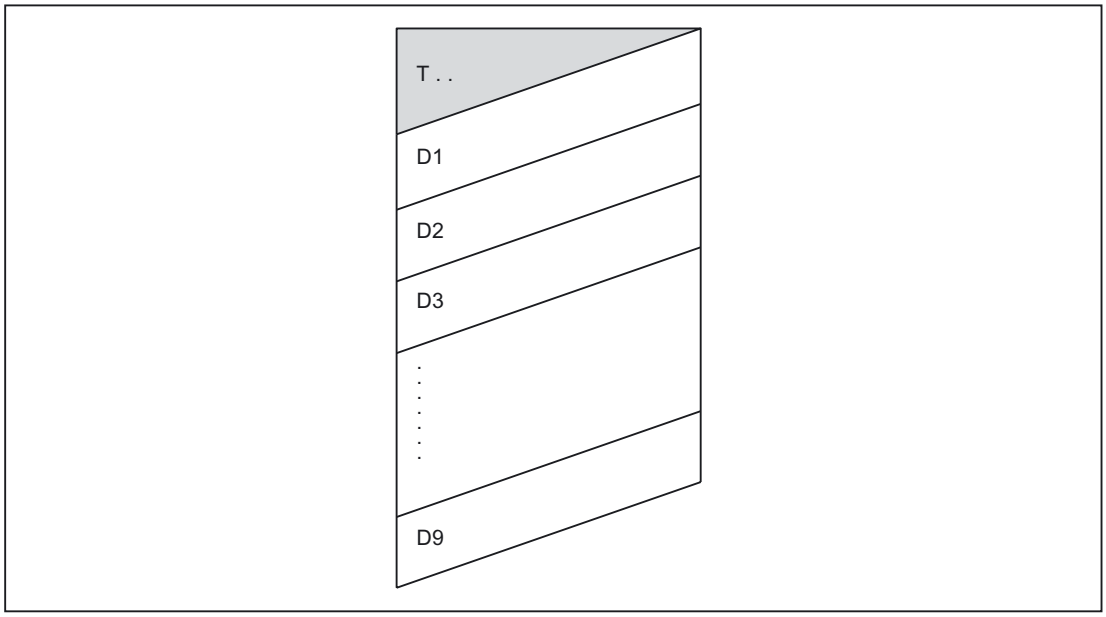


Figure 17-1 Example of a tool T... with 9 cutting edges (D1 to D9)

D function

The tool cutting edge is programmed with D1 (edge 1) to D9 (edge 9). The tool cutting edge always refers to the currently active tool. An active tool cutting edge (D1 to D9) without an active tool (T0) is inactive. Tool cutting edge D0 deselects all tool compensations of the active tool.

Selection of the cutting edge when changing tool

When a new tool (new T number) has been programmed and the old one replaced, the following options are available for selecting the cutting edge:

- The cutting edge number is programmed.
- The cutting edge number is defined by the machine data:

  MD20270 $MC_CUTTING_EDGE_DEFAULT (basic position of the tool cutting edge without programming)

Value	Meaning
= 0	No automatic cutting edge selection in accordance with M06
< > 0	Number of the cutting edge, which is selected in accordance with M06
= -1	The cutting edge number of the old tool is retained and is also selected for the new tool, in accordance with M06.

Activating the tool offset

D1 to D9 activate the tool compensation for a cutting edge on the active tool. Tool length compensation and tool radius compensation can be activated at different times:

- **Tool length compensation** (TLC) is performed on the first traversing motion of the axis, on which the TLC is to act.

  This traversing motion must be a linear interpolation (G0,G1, POS,POSA) or polynomial interpolation (POLY). If the POS/POSA axis is one of the active geometry axes, the tool length compensation is applied with the first axis motion in which the WLK is supposed to act.

- **Tool radius compensation** (TRC) becomes active when G41/G42 is programmed in the active plane (G17, G18 or G19).

  The selection of tool radius compensation with G41/G42 is only permitted in a program block with G0 (rapid traverse) or G1 (linear interpolation).

17.2.2 Compensation memory structure

Tool compensation memory size

Each channel can have a dedicated tool compensation memory (TO unit).

Which tool compensation memory exists for the relevant channel is set with the machine data:

MD28085 $MC_MM_LINK_TOA_UNIT (Assignment of TO unit to a channel).

The maximum number of tool cutting edges for all tools managed by the NCK is set with the machine data:

MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA (number of tool cutting edges in NCK).

Tools

The TO memory consists of tools numbered T1 to T32000.

Each tool can be set up via TOA files or individually, using the "New tool" soft key. Compensation values not required must be assigned the value zero. (this is the default setting when the offset memory is created): The individual values in the offset memory (tool parameters) can be read and written from the program using system variables.

Note

The tools (T1 to T32000) do not have to be stored in ascending order or contiguously in the tool compensation memory, and the first tool does not have to be assigned number T1.

Tool cutting edges

Each tool can have up to 9 cutting edges (D1 to D9). The first cutting edge (D1) is set up automatically when a new tool is loaded in the tool compensation memory. Other cutting edges (up to 8) are set up consecutively and contiguously using the "New cutting edge" soft key. A different number of tool cutting edges can assigned to each tool in this way.

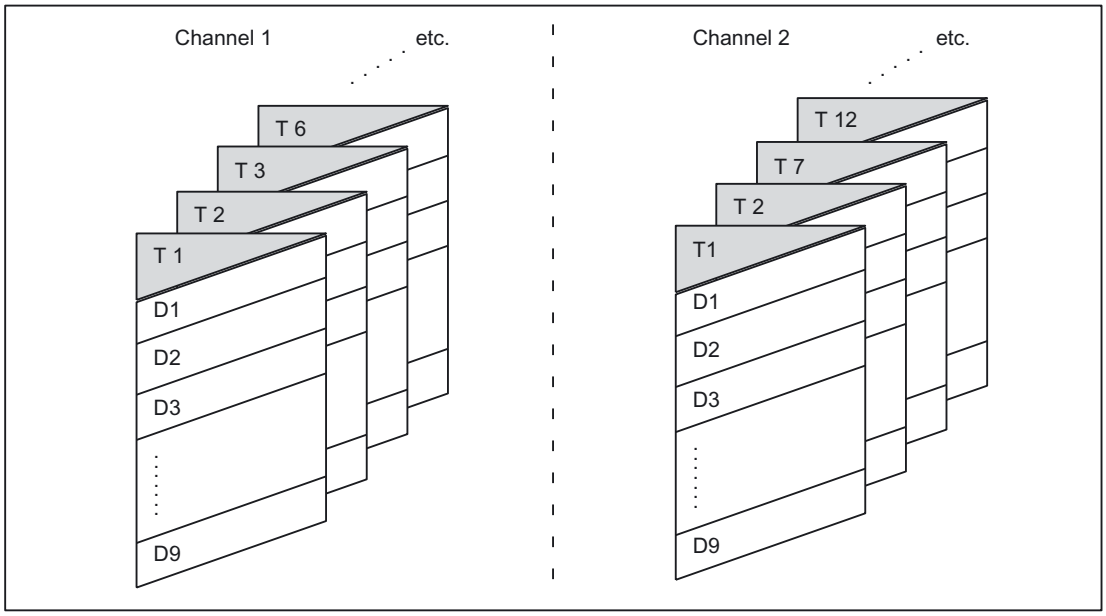
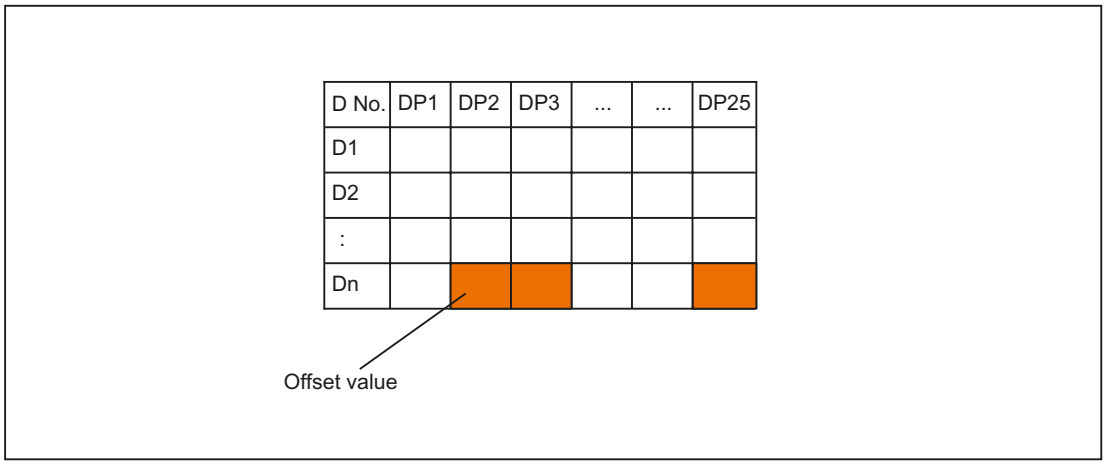


Figure 17-2 Example of a tool compensation memory structure for 2 channels

17.2.3 Calculating the tool compensation

D No.

The **D no.** is sufficient for calculating the tool compensations (can be set via MD).



Programming

The above compensation block is to be calculated in the NC.

Part program call:

```
...
Dn
```

17.2.4 Address extension for NC addresses T and M

MD20096

Whether also with tool management **not** activated, the address extension of T and M is to be interpreted as spindle number, can be set through the machine data:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO (spindle number as address extension).

The same rules then apply to the reference between the D number and T number as when the "Tool management" function is active.

Effect on the D number

A compensation data set is determined by the D number.

The D address cannot be programmed with an address extension.

The evaluation of the D address always refers to the currently active tool.

The programmed D address refers to the active tool in relation to the master spindle (same as for tool management function), when machine data is set:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = TRUE (spindle number as address extension).

Effect on the T number

If the "Tool management" function is active, the values programmed with reference to the master spindle (or master toolholder) are displayed as programmed/active T numbers.

If tool management is not active, **all** programmed T values are displayed as programmed/active, regardless of the programmed address extension.

Only the T value programmed in relation to the master spindle is shown as programmed/active, when:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = TRUE (spindle number as address extension).

Example

The example below shows the effect of MD20096.

Two spindles are considered. Spindle 1 is the master spindle. `M6` was defined as the tool change signal.

```
T1 = 5
M1 = 6
T2 = 50
M2 = 6
D4
```

- If tool management is active, D4 refers to tool "5".

  T2=50 defines the tool for the secondary spindle, whose tool does not influence the path compensation. The path is determined exclusively by the tool programmed for the master spindle.

- D4 relates to tool "50" without active tool management and with the machine data:

  MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = **FALSE** (significance of address extension on T, M tool change).

  The address extensions of neither T nor M are evaluated in the NCK.

  Each tool change command defines a new path compensation.

- D4 relates to tool "5" (as when tool management is active) without active tool management and with the machine data:

  MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = **TRUE**.

  Address extension 1 (T1= ..., M1= ...) addresses the master spindle.

Note

Previously, when tool management was not activated, each tool change command (programmed with T or M) caused the tool compensation to be recalculated in the path. The address extension is not defined further by this operation. The significance of the extension is defined by the user (in the PLC user program).

17.2.5 Free assignment of D numbers

"Relative" D numbers

In the NCK, it is possible to manage the D numbers as "relative" D numbers for the tool compensation data sets. The corresponding D numbers are assigned to each T number. The maximum number of D numbers was previously limited to 9.

Functions

Expansions to functions when assigning D numbers:

- The maximum permitted D numbers are defined via the machine data:

  MD18105 $MN_MM_MAX_CUTTING_EDGE_NO (max. value of the D numbers (DRAM))

  The default value is 9, in order to maintain compatibility with existing applications.

- The number of cuts (or the offset data sets) **for each tool** can be defined via the machine data:

  MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL (max. number of the D numbers per tool (DRAM))

  This allows you to customize the number of cutting edges to be configured for each tool to the actual number of real cutting edges for monitoring purposes.

- It is also possible to rename D numbers in the NCK and thus to allocate any D numbers to the cutting edges.

Note

In addition to relative D number allocation, the D numbers can also be assigned as "flat" or "absolute" D numbers (1-32000) without a reference to a T number (within the "Flat D number structure" function).

Cutting edge number CE

When you rename D numbers, the information in the tool Catalog detailing the numbers defined for these cutting edges is lost. It is, therefore, impossible to determine, following renaming, which cutting edge of the Catalog is being referenced.

Since this information is required for retooling procedures, a **cutting edge number CE** has been introduced for each cutting edge. This number remains stored when the D number is renamed.

The D number identifies the cutting edge compensation in the part program. This **compensation number D** is administered separately from the **cutting edge number CE** (the number in the tool Catalog). Any number can be used. The number is used to identify a compensation in the part program and on the display.

The CE number identifies the actual physical cutting edge during retooling. The cutting edge number CE is not evaluated by the NCK on compensation selection during a tool change (only available via the OPI).

The cutting edge number CE is defined with system variable **$TC_DPCE[t,d]**:

- **t** stands for the internal T number.
- **d** stands for the D number.

Write accesses are monitored for collisions, i.e. all cutting edge numbers of a tool must be different. The variable $TC_DPCE is a component of the cutting edge parameter data set $TC_DP1 to $TC_DP25.

It is only practical to parameterize $TC_DPCE if the maximum cutting edge number (MD18105) is greater than the maximum number of cutting edges per tool (MD18106).

In this case, the default cutting edge number is the same as the classification number of the cutting edge. Compensations of a tool are created starting at number 1 and are incremented up to the maximum number of cutting edges per tool (MD18106).

The cutting edge number CE is the same as the D number (in compatibility with the behavior till now) if:

MD18105 ≤ MD18106.

A read operation returns CE=D. A write operation is ignored without an alarm message.

Note

The compensation values $TC_DP1 to $TC_DP25 of the active tool compensation can be read with system variable $P_AD[n], where n=1 to 25. The CE cutting edge number of the active compensation is returned with n=26.

Commands

When the maximum cutting edge number:
MD18105 $MN_MM_MAX_CUTTING_EDGE_NO (Max. value of the D numbers (DRAM))
is greater than the maximum number of cutting edges per tool:
MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL (Max. number of D numbers per tool (DRAM))
the following commands are available:

Command	Meaning
CHKDNO	Checks the uniqueness of the available D numbers. The D numbers of all tools defined within a TO unit may not occur more than once. No allowance is made for replacement tools.
GETDNO	Determines the D number for the cutting edge of a tool. If no D number matching the input parameters exists, d=0. If the D number is invalid, a value greater than 32000 is returned.
SETDNO	Sets or changes the D number of the CE cutting edge of tool T. If there is no data block for the specified parameter, the value FALSE is returned. Syntax errors generate an alarm. The D number cannot be set explicitly to 0.
GETACTTD	Determines the associated T number for an absolute D number. There is not check for uniqueness. If several D numbers within a TO unit are the same, the T number of the first tool found in the search is returned. This command is not suitable for use with "flat" D numbers, because the value 1 is always returned in this case (no T numbers in database).
DZERO	Marks all D numbers of the TO unit as invalid. This command is used for support during retooling. Compensation data sets tagged with this command are no longer verified by the CHKDNO language command. These data sets can be accessed again by setting the D number once more with SETDNO.

Note

If the maximum cutting edge number is smaller than the maximum number of cutting edges per tool (**MD18105 < MD18106**), the language commands described do not affect the system.

This relation is preset in the NCK as standard, in order to maintain compatibility with existing applications.

The individual commands are described in detail in:

References:
Programming Manual, Fundamentals

Activation

In order to work with unique D numbers and, therefore, with the defined language commands, it must be possible to name D numbers freely for the tools.

The following conditions must be fulfilled for this purpose:

- MD18105 > MD18106
- The 'flat D number' function is not activated.

  MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE (type of D number programming (SRAM)).

Examples

MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 1 (max. value of the D numbers (DRAM))

A maximum of one compensation can be defined per tool (with D number = 1).

Note

When the "Flat D numbers" function is active, only one D compensation can be defined in the TO unit.

MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 9999

Tools can be assigned unique D numbers.

For example:

- D numbers 1, 2, 3 are assigned to T number 1
- D numbers 10, 20, 30, 40, 50 are assigned to T number 2
- D numbers 100, 200 are assigned to T number 3
- etc.

`CHKDNO`; **MD18105 $MN_MM_MAX_CUTTING_EDGE_NO =** 9999

The following data are to be checked for unique D numbers:

- T number 1 with D numbers 1, 2, 3
- T number 2 with D numbers 10, 20, 30, 40, 50
- T number 3 with D numbers 100, 200, 30

  (typing error during definition: 30 was entered instead of 300)

CHKDNO	The FALSE state is returned when the above constellation is checked because D=30 has been entered twice.
CHKDNO (2, 3, 30)	The FALSE state is returned when the specified D number 30 is checked because D=30 has been entered twice.
CHKDNO (2, 3, 100)	The TRUE state indicates that D=100 has been entered just once.
CHKDNO (1, 3)	The TRUE state is returned although there is a conflict between the D=30 of the third tool and D=30 of the second tool.

MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL = 1 (max. number of the D numbers per tool (DRAM))

Only tools with just one cutting edge are used. The value 1 of the machine data inhibits the definition of a second cutting edge for a tool.

MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL = 12

A maximum of 12 cutting edges can be defined per tool.

Programming examples

Renaming a D number

The D number of cutting edge CE = 3 is to be renamed from 2 to 17. The following specifications apply:

- Internal T number T = 1
- D number = 2
- Tool with one cutting edge with:

```
Program code                Comment
$TC_DP2[ 1, 2 ] = 120
$TC_DP3[ 1, 2 ] = 5.5
$TC_DPCE[ 1, 2 ] = 3        ; Cutting edge number CE
```

- MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 20 (max. value of the D numbers (DRAM))

Within the part program, this compensation is programmed as standard with T1, ...D2.

You assign the current D number of cutting edge 3 to a variable (DNoOld) and define the variable DNoNew for the new D number:

```
Program code
def int DNoOld, DNoNew = 17
DNoOld = GETDNO( 1, 3 )
SETDNO( 1, 3, DNoNew )
```

The new D value 17 is then assigned to cutting edge CE=3.

Now the data for the cutting edge are addressed via D number 17, both via the system variable and in programming with the NC address D.

This compensation is now programmed in the part program with `T1, ....D17` and the data are addressed as follows:

```
Program code                Comment
$TC_DP2[ 1, 17 ] = 120
$TC_DP3[ 1, 17 ] = 5.5
$TC_DPCE[ 1, 17 ] = 3       ; Cutting edge number CE
```

Note

If a further cutting edge has been defined for the tool, e.g. $TC_DPCE[1, 2] = 1 ; = CE, the D-number 2 of the cutting edge 1 cannot have the same name as the D-number of the cutting edge 3 i.e.: `SETDNO( 1, 1, 17)` returns the status = `FALSE` as return value.

DZERO - Invalidate D numbers

The activation of this command invalidates all D numbers of the tools in the TO unit. It is no longer possible to activate a compensation until valid D numbers are again available in the NCK. The D numbers must be reassigned using the `SETDNO` command.

The following tools must be defined (all with cutting edge number 1):

T1, D1	D no. of cutting edge CE=1
T2, D10	D no. of cutting edge CE=1
T3, D100	D no. of cutting edge CE=1

The following command is then programmed:

```
Program code
DZERO
```

If one of the compensations is now activated (e.g. with T3 D100), an alarm is generated, because D100 is not currently defined.

The D numbers are redefined with:

Program code	Comment
SETDNO(1, 1, 100)	; T=1, cutting edge 1 receives the (new) D number 100
SETDNO(2, 1, 10)	; T=2, cutting edge 1 receives the (old) D number 10
SETDNO(3, 1, 1)	; T=3, cutting edge 1 receives the (new) D number 1

Note

In the event of a power failure, the DZERO command can leave the NCK in an undefined state with reference to the D numbers. If this happens, repeat the DZERO command when the power is restored.

Operating principle of a retooling program

Let us assume you want to ensure that the required tools and cutting edges are available. The status of the tool-holding magazine of the NCK is arbitrary. The D numbers in the part programs for the new machining operation generally do not match the D numbers of the actual cutting edges. The retooling program can have the following appearance:

Program code	Comment
DZERO	; All D numbers of the TO unit are tagged as invalid.
....	; One or more loops over the locations of the magazine(s) to check the tools and their cutting edge numbers. If a tool is found, which is still enabled ($TC_TP8) and has the required cutting edge number CE (GETDNO), the new D number is allocated to the cutting edge (SETDNO).
....	; Loading and unloading operations are performed. It is possible to work with the tool status "to be unloaded" and "to be loaded".
CHKDNO	; Loading/unloading and the operation for renaming D numbers are complete. Individual tools and/or D numbers can be checked, and collisions can be handled automatically according to the return value.

17.2.6 Compensation block in case of error during tool change

MD22550

If a tool preparation has been programmed in the part program and the NCK detects an error (e.g., the data set for the programmed T number does not exist in the NCK), the user can assess the error situation and perform appropriate tasks, in order to subsequently resume machining.

The tool change may be programmed independently, depending on the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool compensation with M function).

MD22550 $MC_TOOL_CHANGE_MODE = 0

```
T= "T no."        ; Tool preparation + tool change in one NC block,
                  ; i.e., when T is programmed a new D compensation becomes
                  ; active in the NCK (see
                  ; machine data MD20270 $MC_CUTTING_EDGE_DEFAULT)
```

MD22550 $MC_TOOL_CHANGE_MODE = 1

```
T= "T no."        ; Tool preparation
M06               ;Change tool
                  ; (the number of the tool-change M code can be changed),
                  ; i.e., when M06 is programmed a new D compensation becomes
                  ; active in the NCK (see
                  ; machine data MD20270 $MC_CUTTING_EDGE_DEFAULT)
```

The following problems can occur if tool management is not active:

- D compensation data set missing
- Error in part program

Note

The "tool not in magazine" problem cannot be detected since the NCK did not have access to any magazine information during tool compensation.

D compensation data set missing

Program execution is interrupted at the block containing the invalid D value (regardless of the value of machine data MD22550). The operator must either correct the program or reload the missing data set.

To do this, he needs the D number for the flat D number function, or otherwise the T number as well. These parameters are transferred when the alarm is triggered.

Error in part program

The options for intervention in the event of an error depend on how the tool change was programmed, defined via the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool compensation with M function).

Tool change with T programming (MD22550 = 0)

In this case, the "Compensation block" function available in the NCK is used. The NC program stops at the NC block in which a programmed T value error was detected. The "Compensation block" is executed again when the program is resumed.

The operator can intervene as follows:

- Correct the part program.
- Reload the missing cutting edge compensation data from the HMI.
- Include the missing cutting edge compensation data in the NCK using "Overstore".

Following operator intervention, the START key is pressed and the block, which caused the error, is executed again. If the error was corrected, the program is continued. Otherwise, an alarm is output again.

Tool change with T and M06 programming (MD22550 = 1)

In this case, an error is detected in the NC block containing the tool preparation (T programming), however this error is to be ignored initially. Processing continues until the tool change request (usually M06) is executed. The program is to stop at this point.

The programmed T address can contain any number of program lines ahead of the M06 command, or the two instructions can appear in different (sub)programs. For this reason, it is not usually possible to modify a block or a compensation block, which has already been executed.

The operator has the same options for intervention as with = 0.

Reloading of missing data is possible. In this case, however, T must be programmed with "Overstore".

If a program error has occurred, the line with the error cannot be corrected (Txx); only the line at which the program stopped and which generated the alarm can be edited. Only when machine data:

MD22562 $MC_TOOL_CHANGE_ERROR_MODE Bit0 = 1 (response on errors in tool change).

The sequence is as follows:

```
Txx               ; Error! Data set with xx does not exist.
                  ; Detect state; detect xx;
                  ; continue in program
....
M06               ; Detect bit memory "xx missing" → output alarm,
                  ; stop program
                  ; Correct block with, e.g., Tyy M06, start,
                  ; block Tyy M06 interpreted and OK.
                  ; Machining continues.
```

The following occurs when this part of the program is executed again:

```
Txx               ; Error! Data set with xx does not exist,
                  ; Detect state; detect xx;
                  ; continue in program
....
Tyy M06           ; Detect bit memory "xx missing" → cancel without further
                  response,
                  ; as Tyy M06 is correct → program does not stop (correct).
```

If necessary, the original point of the T call can be corrected after the end of the program. If the tool change logic on the machine cannot process this, the program must be aborted and the point of the error corrected.

If only one data set is missing, it is transferred to the NCK, Txx is programmed in "Overstore" and the program is subsequently resumed.

As in the case of "missing D number", the required parameter (T number) can be accessed by the user for "missing T number" via the appropriate alarm (17191).

Note

In order to enable program correction, it stops immediately at the faulty Txx block.

The program text operation is also stopped when machine data:

MD22562 $MC_TOOL_CHANGE_ERROR_MODE Bit0=1 (response on errors in tool change).

17.2.7 Definition of the effect of the tool parameters

MD20360

The effect of the tool parameters on the transverse axis in connection with diameter programming can be controlled selectively with the machine data:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).

Details are described with the mentioned MD

DRF handwheel traversal with half distance

During DRF handwheel traversal, it is possible to move a transverse axis through only half the distance of the specified increment as follows:

Specify the distance with handwheel via the machine data:

MD11346 $MN_HANDWHEEL_TRUE_DISTANCE = 1 (handwheel path or speed specification)

Define the DRF offset in the transverse axis as a diameter offset with the machine data:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK bit 9 = 1 (definition of tool parameters)

Deselecting an axial DRF compensation (DRFOF) also deletes an existing tool compensation (handwheel override in tool direction).

Note

For further information about superimposed movements with the handwheel, please refer to:

References:
Function Manual, Extended Functions; Manual and Handwheel Travel (H1)

Programming Manual, Fundamentals

(The Programming Manual describes the complete technical program options in order to deselect the DRF offset axis-specifically.)

17.3 Flat D number structure

17.3.1 General

Simple tool management

Simple tool management (no replacement tools, no magazines) using D numbers is possible for turning machines.

The function is available in the basic level of tool management (without tool management function activated). Grinding tools cannot be defined using this function.

Activation

Which type of D number management is valid may be set via the machine data:

MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE (type of D number programming).

Value	Significance
0	As previously = default setting
1	Flat D number structure with absolute **direct D programming**

Cutting edges can be deleted individually via PI command or NC programming command. Cutting edges with a specific number can also be created selectively using HMI.

17.3.2 Creating a new D number (compensation block)

Programming

Tool compensations can be programmed with system variables $TC_DP1 to $TC_DP25. The contents have the same meaning as before.

The syntax changes: no T number is specified.

- "Flat D number" function **active**:

  $TC_DPx[d] = value ;where x=parameter no., d=D number

  i.e., data with this syntax can only be loaded to the NCK if the "Flat D number" function is activated.

- "Flat D number" function **inactive**:

  $TC_DPx[t][d] = value ;where t=T number, d=D number

A D number can only be assigned once for each tool, i.e., each D number stands for exactly one compensation data block.

A new data block is stored in the NCK memory when a D number that does not exist is created for the first time.

The maximum number of D or offset data blocks (max. 600) is set via the machine data:

MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA (tool compensations in TO area).

Data backup

Data backup is carried out in the same format, i.e., a backup file created with the "Flat D number" function cannot be loaded on the NCK of a control that has not activated the function.

This also applies in reverse for transfer.

D range

1 - 99 999 999

17.3.3 D number programming

MD18102 = 1

If MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE = 1, then D compensation is activated without reference to a certain tool.

`D0` still contains the previous significance, "Deselection of active compensation in NCK".

Address extension of D

It is not possible to extend the address of D. Only one active compensation data block is possible for the tool path at a given time.

Programming

Programming in the part program is carried out as before. Only the value range of the programmed D number is increased.

Example 1:

MD parameterization		Meaning
MD22550 $MC_TOOL_CHANGE_MODE	= 0	Tool change with programming of T.
MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE	= 1	D number programming without reference to a certain tool.
MD20270 $MC_CUTTING_EDGE_DEFAULT	= -1	D compensation remains unchanged if tool is changed.

```
Program code                Comment
...
D92
X0                          ; Traverse with compensations from D92.
T17                         ; Outputs T=17 to the PLC
X1                          ; Traverse with compensations from D92.
D16
X2                          ; Traverse with compensations from D16.
D32000
X3                          ; Traverse with compensations from D32000.
T29000                      ; Outputs T=29000 to the PLC.
X4                          ; Traverse with compensations from D32000.
D1
X5                          ; Traverse with compensations from D1.
...
```

Example 2:

MD22550 = 0

```
Program code                Comment
T1
T2
T3
D777                        ; No waiting, D777 is activated, T3= programmed and
                              active tool in the display, D777= programmed and
                              active compensation.
```

Note

The tool change and the assignment of a D compensation to an actual tool are the responsibility of the NC program and of the PLC program, if applicable.

Delete D no. via part program

- **With** flat D number:

  $TC_DP1[d] = 0

  Compensation data block with the number D in the TO unit is deleted.

  The memory is then free for the definition of another D number.

- **Without** flat D number:

  $TC_DP1[t][d] = 0

  Cutting edge d of tool t is deleted.

- $TC_DP1[0] = 0

  All D compensations of the TO unit are deleted.

Active compensation data blocks (D numbers) cannot be deleted. This means, that it may be necessary to program `D0` before deleting.

Tool MDs

The following machine data affect the way tools and cutting edges (D numbers) work in the NCK:

Machine data	Meaning
MD20270 $MC_CUTTING_EDGE_DEFAULT	Standard tool cutting edge after tool change
MD20130 $MC_CUTTING_EDGE_RESET_VALUE	Tool cutting edge - Length compensation on power-up (RESET / TP end)
MD20120 $MC_TOOL_RESET_VALUE	Tool - Length compensation on power-up (RESET / TP end)
MD20121 $MC_TOOL_PRESEL_RESET_VALUE	Preselect tool on RESET
MD22550 $MC_TOOL_CHANGE_MODE	Tool change with M function instead of T function
MD22560 $MC_TOOL_CHANGE_M_CODE	M function for tool change
MD20110 $MC_RESET_MODE_MASK	Definition of basic control settings after RESET/TP-End)
MD20112 $MC_START_MODE_MASK	Determination of basic control settings after NC start

17.3.4 Programming the T number

When the "Flat D number structure" function is active, NC address T continues to be evaluated, i.e., the programmed T number and the active T number are displayed. However, the NC determines the D number without reference to the programmed T value.

The NC detects 1 master spindle per channel (via the spindle number, which can be set using MD). Compensations and the `M6` command (tool change) are only calculated in reference to the master spindle.

An address extension T is interpreted as a spindle number (e.g., T2 = 1; tool 1 to be selected on spindle 2); a tool change is only detected if spindle 2 is the master spindle.

17.3.5 Programming M6

MD22550 and MD22560

The NC detects 1 master spindle per channel (via the spindle number, which can be set using MD). Compensations and the `M6` command (tool change) are only calculated in reference to the master spindle.

Whether the tool change command is performed with an M function is defined via the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool compensation with M function).

T is used as the tool preparation command.

The name of the M function for the tool change is defined via the machine data:

MD22560 $MC_TOOL_CHANGE_M_CODE (M function for tool change).

The default is M6. An address extension of M6 is interpreted as a spindle number.

Example

Two spindles are defined, spindle 1 and spindle 2, and the following applies:

```
MD20090 = 2       ; Spindle no. 2 is the master spindle.
M6                ; Tool change desired, command refers implicitly to the master
                  spindle
M1 = 6            ; No tool change, since spindle no. 2 is the master spindle
M2 = 6            ; Tool is changed, since spindle no. 2 is the master spindle
```

17.3.6 Program test

MD20110

To have the active tool and the tool compensation included as follows, can be defined via the machine data:

MD20110 $MC_RESET_MODE_MASK, Bit 3 (Definition of control default settings after RESET/TP end).

Value		Significance
Bit 3	= 1	From the last test program to finish in test mode
	= 0	From the last program to finish before activation of the program test

Prerequisite

The bits 0 and 6 must be set by the machine data:

MD20110 $MC_RESET_MODE_MASK, Bit 3 (Definition of control default settings after RESET/TP end).

17.3.7 Tool management or "Flat D numbers"

Tool management

NCK active tool management works on the basis of the following assumptions:

1. Tools are managed in magazines.
2. Cutting edges are monitored; limits reached cause the tool to be disabled.
3. Idea behind replacement tools: Tools are programmed for selection only the on the basis of their identifier. NCK then selects the concrete tool according to the strategy.

Note

For SINUMERIK 828D, this function is only available as option!

This means that it only makes sense to employ tool management when specific tools have been defined and these are to be utilized by the NCK.

Flat D number

Flat D number means that tool management is carried out outside the NCK and there is no reference made to T numbers.

No mixture of tool management and flat D no.

It does not make sense to mix or distribute the tool management functions over the NCK and PLC, since the main reason for tool management on the NCK is **to save time**.

This only works if the tasks that are timecritical are carried out on the NCK. This is not the case for "Flat D number", however.

Note

Activation of both of the functions "Flat D number structure" and "Tool management" is monitored. If both are activated at the same time, "Tool management" takes priority.

17.4 Tool cutting edge

17.4.1 General

Tool cutting edge

The following data are used to describe a tool cutting edge uniquely:

- Tool type (end mill, drill, etc.)
- Geometrical description
- Technological description

Tool parameter

The geometrical description, the technological description and the tool type are mapped to tool parameters for each tool cutting edge.

The following tool parameters are available for the relevant tool types:

Tool parameter	Significance	Note
1	Tool type	
2	Cutting edge position	for turning tools or for milling/grinding tools with 2D TRC contour tool
Geometry - tool lengths		
3	Length 1	
4	Length 2	
5	Length 3	
Geometry - tool shape		
6	Radius 1/Length 1	for 3D face milling
7	Length 2	for 3D face milling
8	Radius 1	for 3D face milling
9	Radius 2	for 3D face milling
10	Angle 1 / minimum threshold angle	for 3D face milling with 2D TRC contour tool
11	Angle 2 / minimum threshold angle	for 3D face milling with 2D TRC contour tool
Wear - tool length		
12	Length 1	
13	Length 2	
14	Length 3	

Tool parameter	Significance	Note
Wear - tool shape		
15	Radius 1/Length 1	for 3D face milling
16	Length 2	for 3D face milling
17	Radius 1	for 3D face milling
18	Radius 2	for 3D face milling
19	Angle 1 / minimum limit angle	for 3D face milling with 2D TRC contour tool.
20	Angle 2 / maximum limit angle	for 3D face milling with 2D TRC contour tool.
Tool base dimension / adapter dimension		
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 is not used and is reserved for expansions.

3D-face milling

Milling cutter types 111, 120, 121, 130, 155, 156 and 157 are given special treatment for 3D-face milling by evaluating tool parameters (1 -23).

References

For more information about various tool types, see:

- Functions Manual - Basic Functions; Tool Offset (W1), Chapter "Tool type (tool parameters)"
- Programming Manual Fundamentals; Chapter: "Tool compensations" > "List of tool types"
- Functions Manual - Special Functions; 3D-tool radius compensation (W5)

17.4.2 Tool parameter 1: Tool type

Description

The tool type (3digit number) defines the tool in question. The selection of this tool type determines further components such as geometry, wear and tool base dimensions in advance.

Conditions

The following is applicable to the "Tool type" parameter:

- The tool type must be specified for each tool cutting edge.
- Only the values specified can be used for the tool type.
- Tool type "0" (zero) means that no valid tool has been defined.

Tool types and tool parameters

Different tool types and the most important tool parameters are listed in the following table. The tool parameters available for a certain tool type are designated with "x".

Table of the most important available tool paramaters (only tool type necessary)

1	tool parameter $TC_DP	2	3	4	5	6	7	8/9	12/13/14	15/16	21/22/23	24	TPG 3	TPG 4/5	TPG 6	TPG 7	TPG 8
tool type	compensation by NCK	cutting edge position	geometry - length 1	– length 2	– length 3	geometry - radius 1	slot width / - radius 2	projection / - length 4/5	wear - length 1/2/3	wear - radius 1/2	adap/base measure. - length 1/2/3	tool clearance angle	– disc radius min	– disc width min/act.	– speed max	– peripheral speed max	– angle gradient disc
Milling tools and drilling tools (all others)																	
100	Milling tools according to CLDATA*1		x	x	x	x			x	x	x						
110	ball end mill		x	x	x	x			x	x	x						
120	end mill		x	x	x	x			x	x	x						
121	end mill with corner rounding		x	x	x	x	x		x	x	x						
130	angle head cutter		x	x	x	x			x	x	x						
131	angle head cutter with corner rounding		x	x	x	x	x		x	x	x						
140	face mill		x	x	x	x			x	x	x						
145	thread mill		x	x	x	x			x	x	x						
150	side mill		x	x	x	x			x	x	x						
155	truncated cone mill		x	x	x	x			x	x	x						
200	twist drill		x	x	x	x			x		x						
205	solid bit		x	x	x	x			x		x						
210	drilling rod		x	x	x	x			x		x						
220	center drill		x	x	x	x			x		x						
230	countersink		x	x	x	x			x		x						
231	piloted counterbore		x	x	x	x			x		x						
240	tap drill regular thread		x	x	x	x			x		x						
241	tap drill fine thread		x	x	x	x			x		x						
242	tap drill Whitworth thread		x	x	x	x			x		x						
250	reamer		x	x	x	x			x		x						
grinding tools and turning tools (400 - 599)																	
400	peripheral grinding wheel	x	x	x	x	x			x	x	x					x	x
401	peripheral grinding wheel with monitoring	x	x	x	x	x			x	x	x		x	x	x	x	x
403	as 401 but without base measurement for SUG*2	x	x	x	x	x			x	x	x		x	x	x		x
410	face plate	x	x	x	x	x			x	x	x					x	
411	face plate with monitoring	x	x	x	x	x			x	x	x		x	x	x	x	
413	as 411 but without base measurement for SUG*2	x	x	x	x	x			x	x	x		x	x	x		
490	dresser	x	x	x	x	x			x	x	x						
500	roughing tool	x	x	x	x	x			x	x	x	x					
510	smoothing tool	x	x	x	x	x			x	x	x	x					
520	recessing tool	x	x	x	x	x			x	x	x	x					
530	cutting-off tool	x	x	x	x	x			x	x	x	x					
540	threading tool	x	x	x	x	x			x	x	x	x					
special tools (700)																	
700	slotting saw		x	x	x	x	x	x	x	x	x						

CLDATA*1 "cutter Location Data" — tool position data according to DIN 66215,
BI 1SUG*2 — grinding wheel peripheral speed

Note

The tool type has no significance in the turning tool groups.

Nonlisted numbers are also permitted, in particular with grinding tools (400-499).

Tool offset data

Tool offset data (TOA data) is stored in the system variables.

Example Slotting saw tool type (Type 700)

	Geometry	Wear	Base	Einheit
Length compensation				
Length 1	$TC_DP3	$TC_DP12	$TC_DP21	mm
Length 2	$TC_DP4	$TC_DP13	$TC_DP22	mm
Length 3	$TC_DP5	$TC_DP14	$TC_DP23	mm
Saw blade compensation				
Diameter d	$TC_DP6	$TC_DP15		mm
Slot width b	$TC_DP7	$TC_DP16		mm
Projection k	$TC_DP8	$TC_DP17		mm

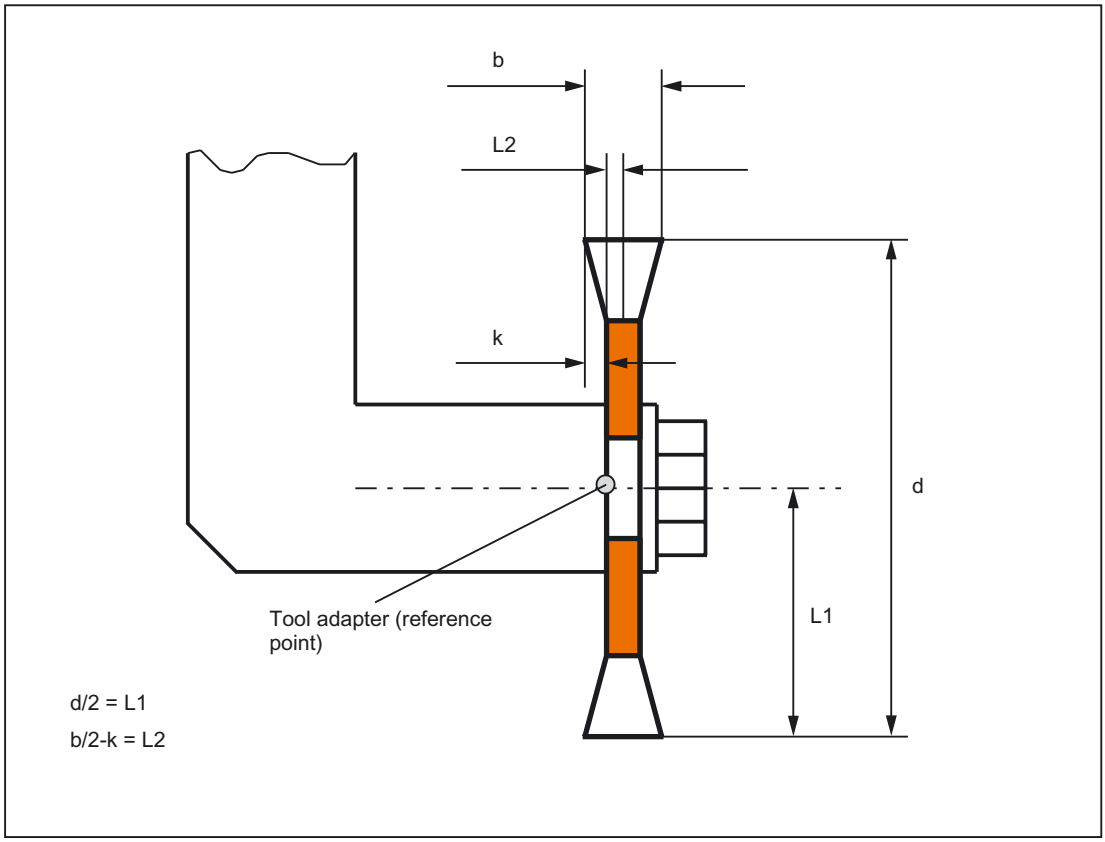


Figure 17-3 Geometry of slotting saw (analogous to angle head cutter)

The width of the saw blade is accounted for with tool radius compensation (G40 to G42 as follows:

Command	Significance
G40	No saw blade compensation
G41	Saw blade compensation left
G42	Saw blade compensation right

17.4.3 Tool parameter 2: Cutting edge position

Description

The cutting edge position describes the position of the tool tip P in relation to the cutting edge center point S. It is entered in tool parameter 2.

The cutting edge position is required together with the cutting edge radius (tool parameter 8) for the calculation of the tool radius compensation for turning tools (tool type 5xx).

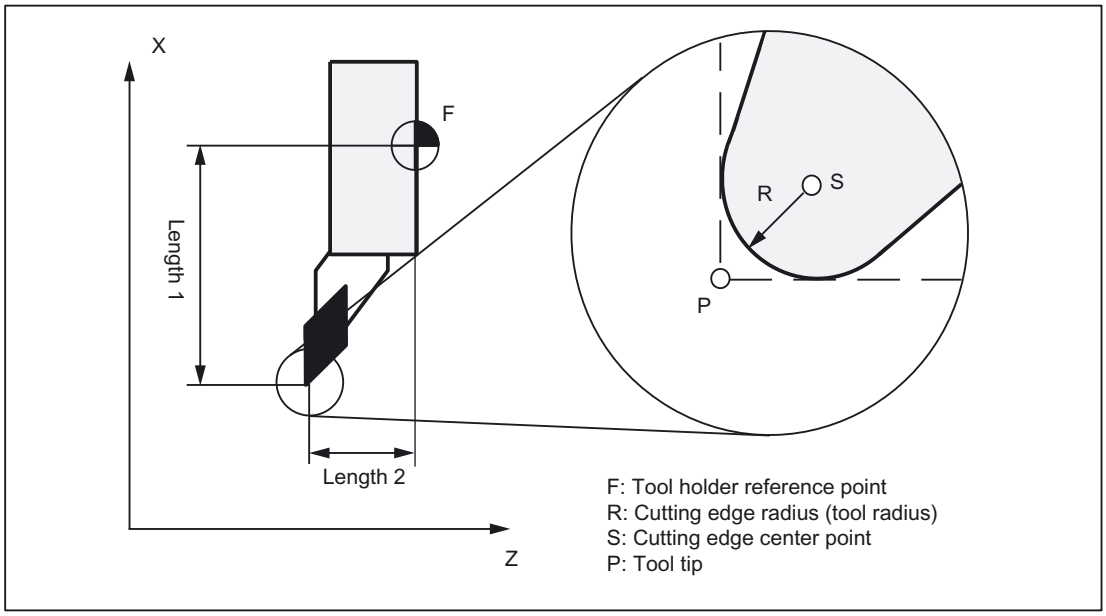


Figure 17-4 Dimensions for turning tools: Turning tool

Cutting edge position parameter values

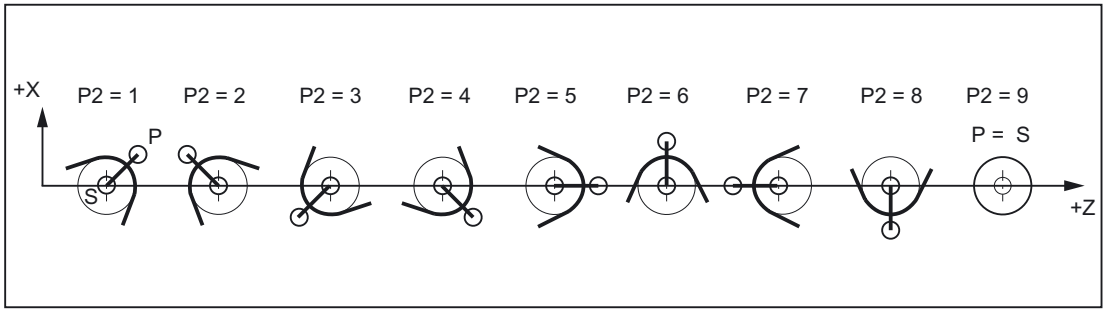


Figure 17-5 Tool parameter 2 (P2): Machining behind the turning center

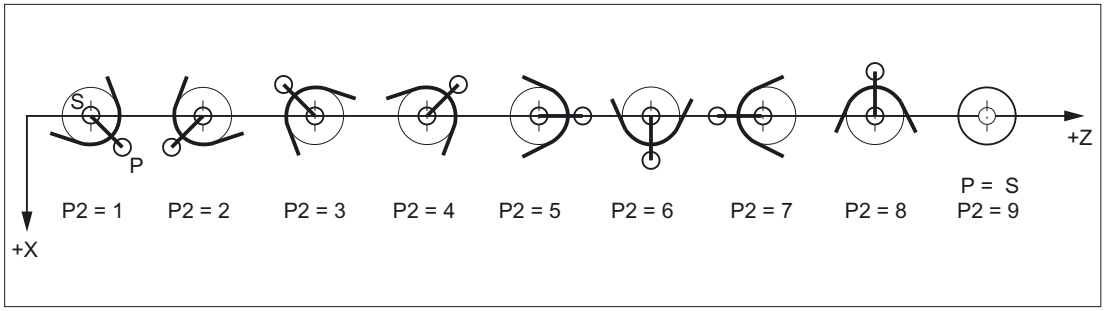


Figure 17-6 Tool parameter 2 (P2): Machining in front of the turning center

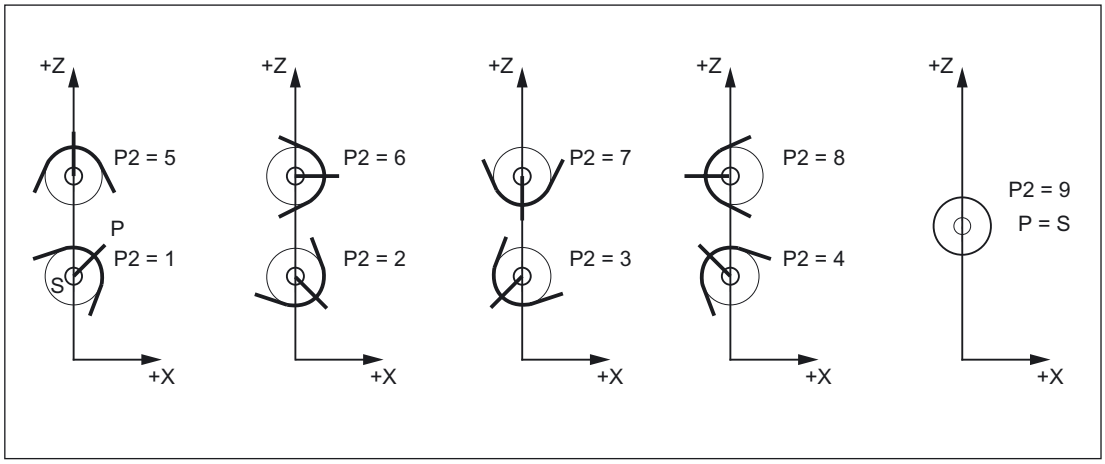


Figure 17-7 Tool parameter 2 (P2): Cutting edge position for vertical boring and turning mills

Special points to be noted

- If the cutting edge center point S is used instead of point P as a reference point to calculate the tool length compensation, the identifier 9 must be entered for the cutting edge position.
- The identifier 0 (zero) is not permitted as a cutting edge position.

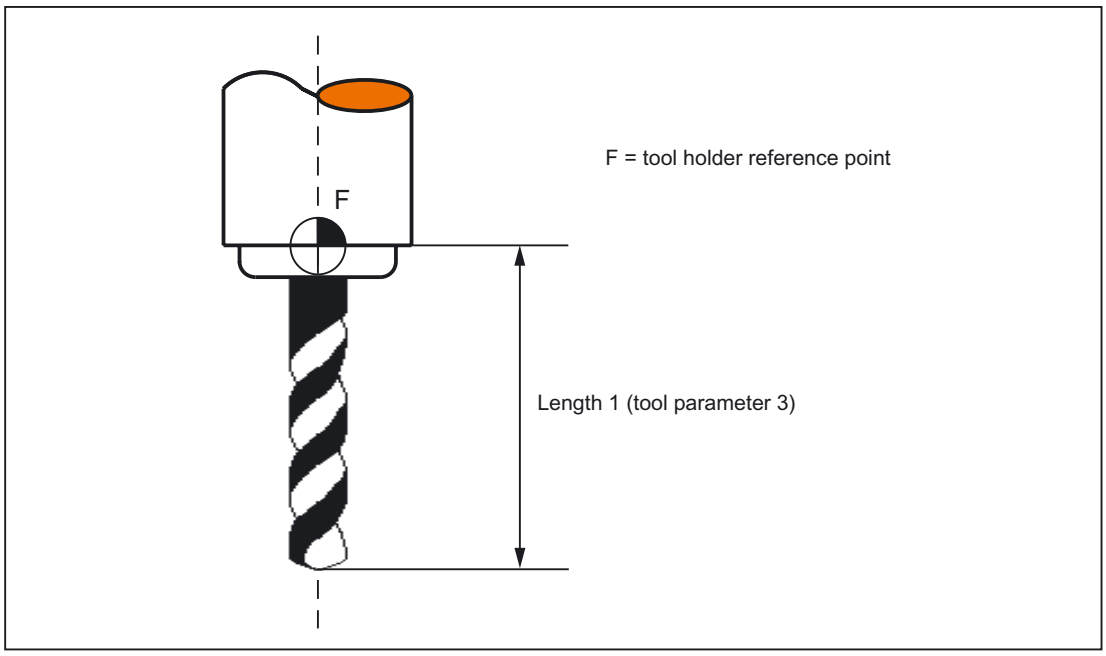
17.4.4 Tool parameters 3 - 5: Geometry - tool lengths

Description

The lengths of the tools are required for the geometry tool length compensation. They are input as tool lengths 1 to 3 in the tool parameters 3 to 5. The following length specifications must be entered as a minimum for each tool type:

Tool type	Required tool lengths
Tool type 12x, 140, 145, 150:	Tool length 1
Tool type 13x:	Tool length 1 to 3 (depending on plane `G17-G19`)
Tool type 2xx:	Tool length 1
Tool type 5xx:	Tool length 1 to 3

Example Twist drill (tool type 200) with tool length (tool parameter 3)



Note

All three tool parameters 3 to 5 (tool length 1 to 3) are always calculated in the three geometry axes, irrespective of the tool type.

If more tool lengths are input in the tool parameters 3 to 5 for a tool type than is required as the minimum, then these extra tool lengths are settled in the geometry axes without any alarm.

Special points to be noted

The active size of the tool is only defined when the geometry tool length compensation (tool parameters 3 to 5) and the wear tool length compensation (tool parameters 12 to 14) are added together. The base-dimension/adapter-dimension tool length compensation is also added in order to calculate the total tool length compensation in the geometry axes.

References

For information about entering tool dimensions (lengths) in tool parameters 3 to 5 (tool lengths 1 to 3) and how these are calculated in the three geometry axes, please refer to → Operating Manual.

17.4.5 Tool parameters 6 - 11: Geometry - tool shape

Meaning

The shape of the tool is defined using the tool parameters 6 to 11. The data is required for the geometry tool radius compensation.

In most cases, only tool parameter 6 (tool radius 1) is used.

Tool parameter	Meaning	Use	
6	Tool length 1	Not used	
7	Tool length 2	Not used	
8	Tool radius 1	The tool radius must be entered for the following tool types in tool parameter 6 (tool radius 1):	
		Tool type 1xx	Milling tools
		Tool type 5xx	Turning tools
		A tool radius does not have to be entered for drilling tools (tool type 2xx). The cutting edge position (tool parameter 2) also has to be entered for turning tools (tool type 5xx).	
9	Tool radius 2	Not used	
10	Tool angle 1	Not used	
11	Tool angle 2	Not used	

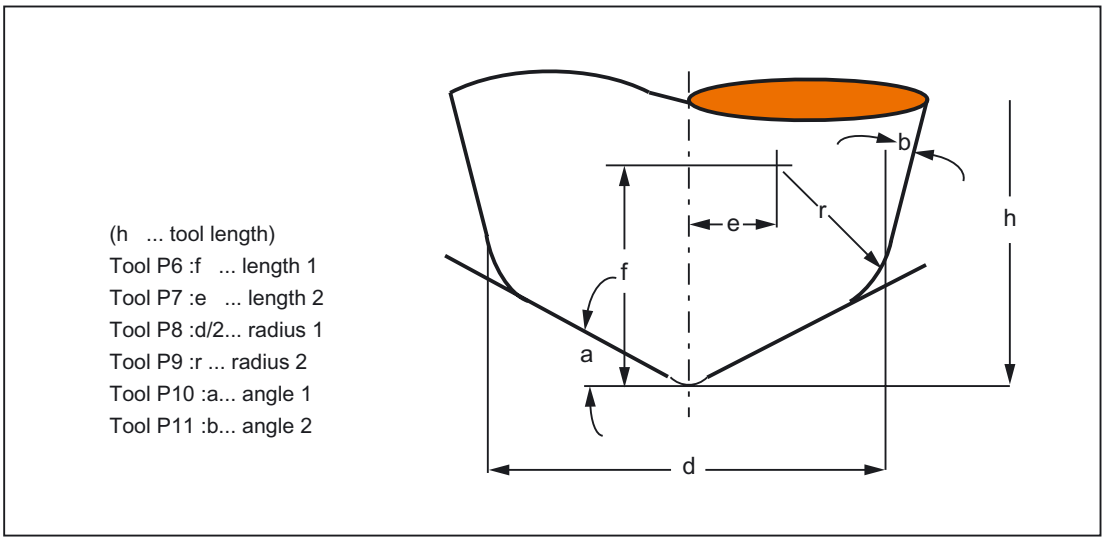
2D TRC with contour tools

For the definition of contour tools with multiple tool cutting edges, the minimum and maximum limit angle can be entered. Both limit angles each relate to the vector of the cutting edge center point to the cutting edge reference point and are counted clockwise.

Tool angle 1	Minimum limit angle per tool cutting edge
Tool angle 2	Maximum limit angle per tool cutting edge

3D face milling

Tool parameters 6 to 11 are required for tool description of 3D face milling.



References

Please refer to the following documentation for information about entering tool shapes (radius for tool radius compensation) in tool parameters 6 to 11 and how these are calculated by geometry tool radius compensation in the three geometry axes:

- Programming Manual, Fundamentals; Section: "Tool compensations" > "2½ D tool compensation"
- Functions Manual - Special Functions; 3D tool radius compensation (W5)

For 3D face milling, please refer to:

- Programming Manual, Job Planning; Section: "Transformations" > "Three, Four and Five axis Transformation (TRAORI)"

17.4.6 Tool parameters 12 - 14: Wear - tool lengths

Description

While geometry tool length compensation (tool parameters 3 to 5) is used to define the size of the tool, wear tool length compensation can be used to correct the change in the active tool size.

The active tool dimensions can change due to:

- Differences between the tool fixture on the tool measurement machine and the tool fixture on the machine tool
- Tool wear caused during service life by machining
- Definition of the finishing allowances

Active tool size

The geometry tool compensation (tool parameters 3 to 5) and the wear tool length compensation (tool parameters 12 to 14) are added together (geometry tool length 1 is added to wear tool length 1, etc.) to arrive at the size of the active tool.

17.4.7 Tool parameters 15 - 20: Wear - tool shape

Description

While geometry tool radius compensation (tool parameters 6 to 11) is used to define the shape of the tool, wear tool radius compensation can be used to correct the change in the active tool shape.

The active tool dimensions can change due to:

- Tool wear caused during service life by machining
- Definition of the finishing allowances

Active tool shape

The geometry tool radius compensation (tool parameters 6 to 11) and the wear tool radius compensation (tool parameters 15 to 20) are added together (geometry tool radius 1 is added to wear tool radius 1, etc.) to arrive at the shape of the active tool.

17.4.8 Tool parameters 21 - 23: Tool base dimension/adapter dimension

Description

Tool base dimension/adapter dimension can be used when the reference point of the toolholder (tool size) differs from the reference point of the toolholder.

This is the case when:

- The tool and the tool adapter are measured separately but are installed on the machine in one unit (the tool size and adapter size are entered separately in a cutting edge).
- The tool is used in a second tool fixture located in another position (e.g. vertical and horizontal spindle).
- The tool fixtures of a tool turret are located at different positions.

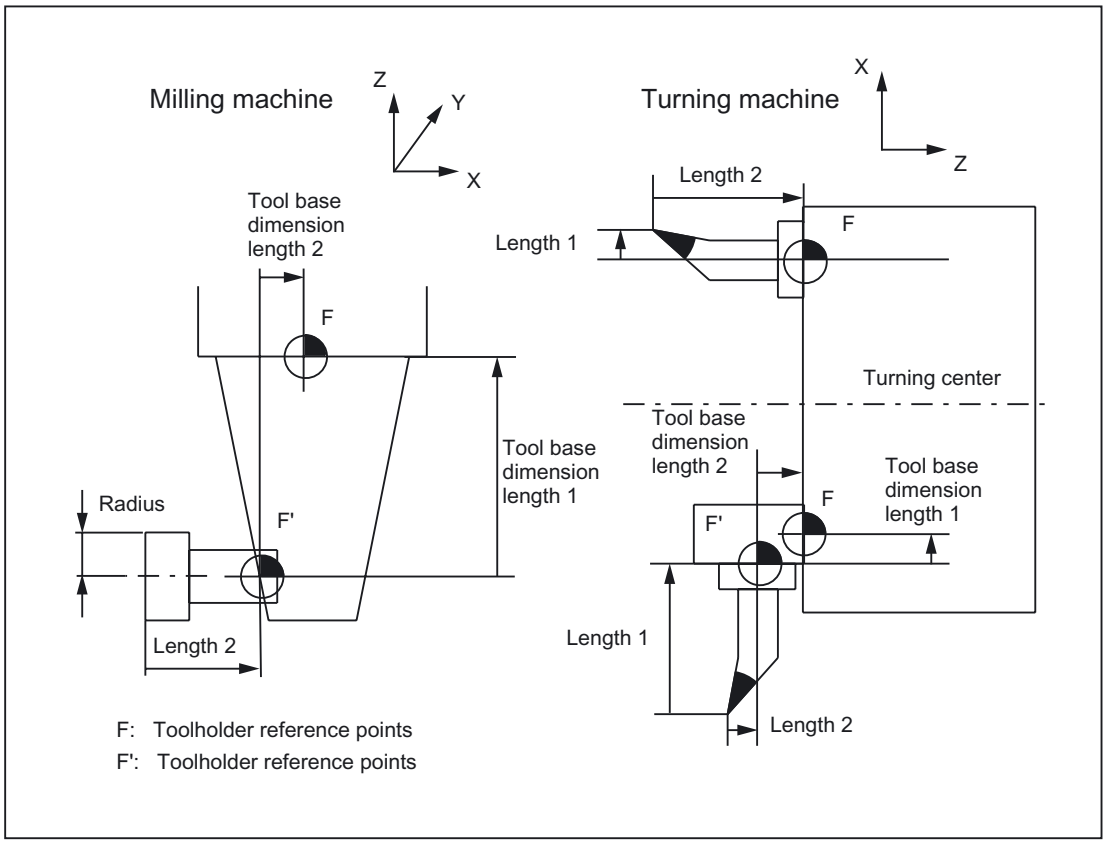


Figure 17-8 Application examples for base-dimension/adapter-dimension TLC

Tool basic length 1 to 3 (tool parameters 21 to 23)

In order that the discrepancy between the toolholder reference point F and the toolholder reference point F' can be corrected on the three geometry axes (three dimensional), all 3 basic lengths are active irrespective of the tool type. In other words, a twist drill (tool type 200) with a tool length compensation (length 1) can also have a tool base dimension/adapter dimension in 3 axes.

References

Please refer to the following documentation for more information about base-dimension/adapter-dimension tool length compensation:

- Programming Manual, Fundamentals

17.4.9 Tool parameter 24: Undercut angle

Meaning

Certain turning cycles, in which traversing motions with tool clearance are generated, monitor the tool clearance angle of the active tool for possible contour violations.

Value range

The angle (0 to 90° with no leading sign) is entered in tool parameter 24 as the tool clearance angle.

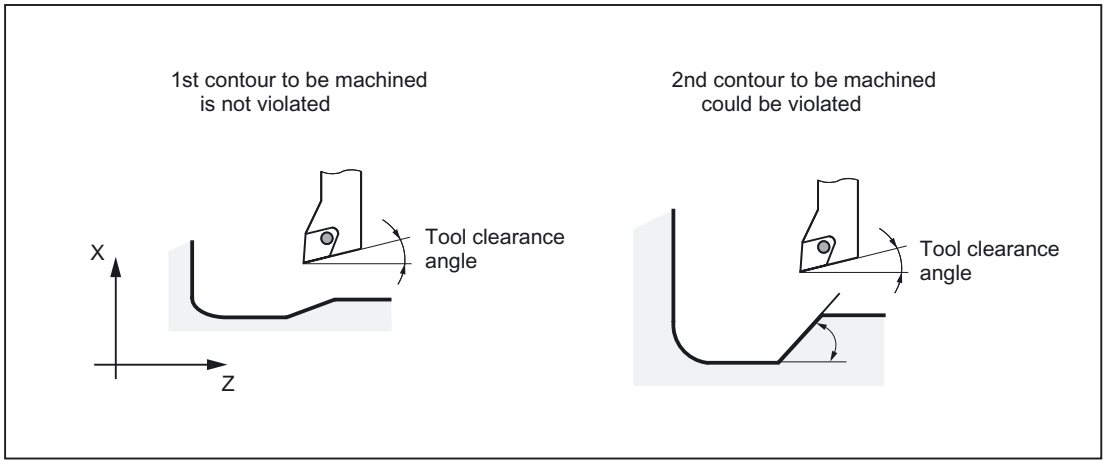


Figure 17-9 Tool clearance angle of the turning tool during relief cutting

Machining type, longitudinal or transverse

The tool clearance angle is entered in different ways according to the type of machining (longitudinal or face). If a tool is to be used for both longitudinal and face machining, two cutting edges must be entered for different tool clearance angles.

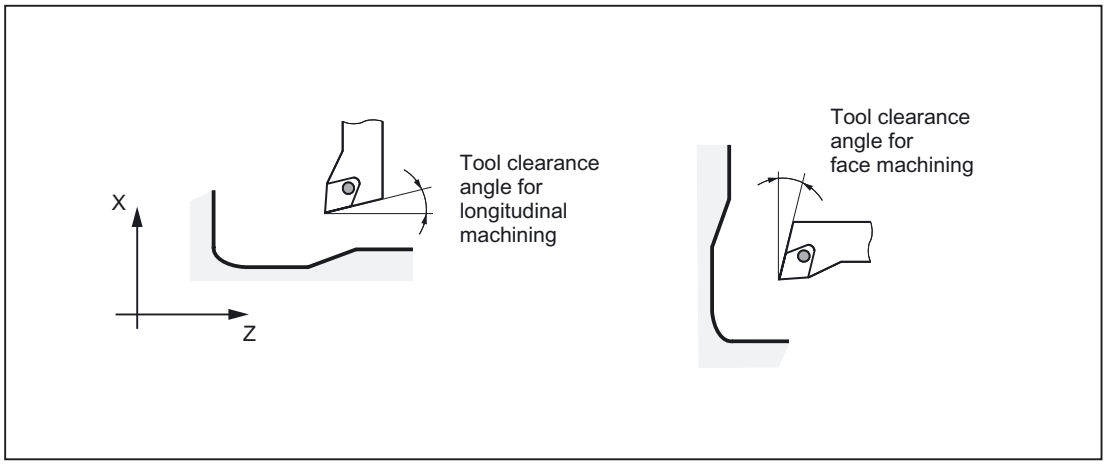


Figure 17-10 Tool clearance angle for longitudinal and face machining

Note

If a tool clearance angle (tool parameter 24) of zero is entered, relief cutting is not monitored in the turning cycles.

References

Please refer to the following documentation for a detailed description of the tool clearance angle:

- Programming Manual Cycles

17.4.10 Tools with a relevant tool point direction

The following must be observed for tools with relevant cutting edge position:

- The straight line between the tool edge center points at the block start and block end is used to calculate intersection points with the approach and retraction block. The difference between the tool edge reference point and the tool edge center point is superimposed on this movement.

  For approach and/or retraction with KONT, the movement is superimposed in the linear subblock of the approach or retraction movement. Therefore, the geometric conditions for tools with or without relevant cutting edge position are identical.

- In circle blocks and in motion blocks containing rational polynomials with a denominator degree > 4, it is not permitted to change a tool with active tool radius compensation in cases where the distance between the tool edge center point and the tool edge reference point changes. With other types of interpolation, it is now possible to change when a transformation is active (e.g. TRANSMIT).

- For tool radius compensation with variable tool orientation, the transformation from the tool edge reference point to the tool edge center point can no longer be performed by means of a simple zero offset. Tools with a relevant cutting edge position are therefore not permitted for 3D peripheral milling (an alarm is output).

Note

The subject is irrelevant with respect to face milling as only defined tool types without relevant cutting edge position are permitted for this operation anyway. (A tool with a type, which has not been explicitly approved, is treated as a ball end mill with the specified radius. A cutting edge position parameter is ignored).

17.5 Tool radius compensation 2D (TRC)

17.5.1 General

Note

For tool radius compensation (TRC) see:
References:
Programming Manual Fundamentals

Only the Programming Guide contains a complete technical description of the tool radius compensation (TRC) and its special aspects.

Why TRC?

The contour (geometry) of the workpiece programmed in the part program should be independent of the tools used in production. This makes it necessary to draw the values for the tool length and tool radius from a current offset memory. Tool radius compensation can be used to calculate the equidistant path to the programmed contour from the current tool radius.

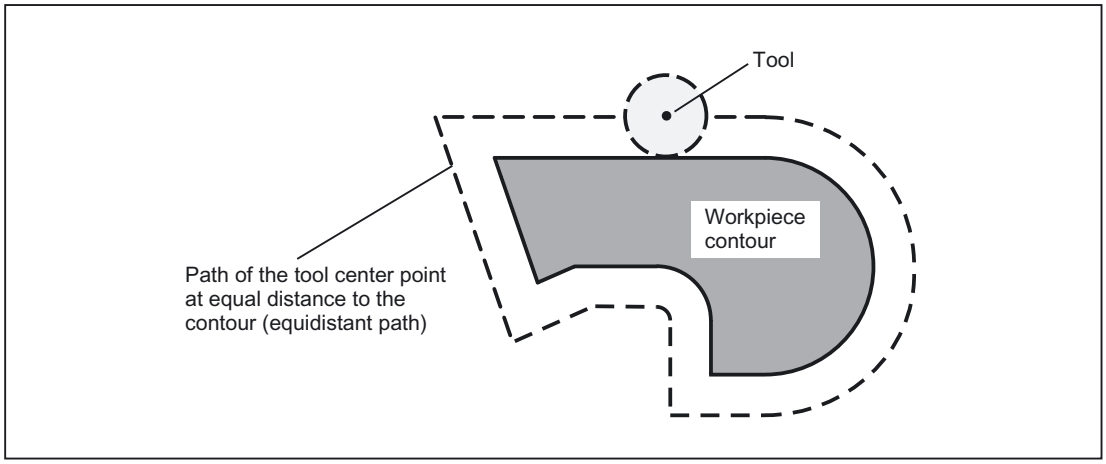


Figure 17-11 Workpiece contour (geometry) with equidistant path

TRC on the plane

TRC is active on the current plane (G17 to G19) for the following types of interpolation:

• Linear interpolation	...	G0, G1
• Circular interpolation	...	G2, G3, CIP
• Helical interpolation	...	G2, G3
• Spline interpolation	...	ASPLINE, BSPLINE, CSPLINE
• Polynomial interpolation	...	POLY

17.5.2 Selecting the TRC (G41/G42)

Direction of compensation

TRC calculates a path, which is equidistant to the programmed contour. Compensation can be performed on the left- or righthand side of the programmed contour in the direction of motion.

Command	Significance
G41	TRC on the lefthand side of the contour in the direction of motion
G42	TRC on the righthand side of the contour in the direction of motion
G40	Deselection of TRC

Intermediate blocks

In general, only program blocks with positions on geometry axes in the current plane are programmed when TRC is active. However, dummy blocks can still also be programmed with active TRC. Dummy blocks are program blocks, which do not contain any positions on a geometry axis in the current plane:

- Positions on the infeed axis
- Auxiliary functions,
- etc.

The maximum number of dummy blocks can be defined in the machine data:

MD20250 $MC_CUTCOM_MAXNUM_DUMMY_BLOCKS (Max no. of dummy blocks with no traversing movements for TRC).

Special points to be noted

- TRC can only be selected in a program block with G0 (rapid traverse) or G1 (linear interpolation).
- A tool must be loaded (T function) and the tool cutting edge (tool compensation) (D1 to D9) activated no later than in the program block with the tool radius compensation selection.
- Tool radius compensation is not selected with a tool cutting edge/tool compensation of D0.
- If only one geometry axis is programmed on the plane when tool radius compensation is selected, the second axis is automatically added on the plane (last programmed position).
- If no geometry axis is programmed for the current plane in the block with the tool radius compensation selection, no selection takes place.
- If tool radius compensation is deselected (G40) in the block following tool radius compensation selection, no selection takes place.
- If tool radius compensation is selected, the approach behavior is determined by the NORM/KONT instructions.

17.5.3 Approach and retraction behavior (NORM/KONT/KONTC/KONTT)

NORM and KONT

The NORM and KONT instructions can be used to control approach behavior (selection of tool radius compensation with G41/42) and retraction behavior (deselection of tool radius compensation with G40):

Command	Significance
NORM	Normal setting at start point/end point (initial setting)
KONT	Follow contour at start point/end point
KONTC	Approach/retraction with constant curvature
KONTT	Approach/retraction with constant tangent

Special points to be noted

- KONT only differs from NORM when the tool start position is behind the contour.

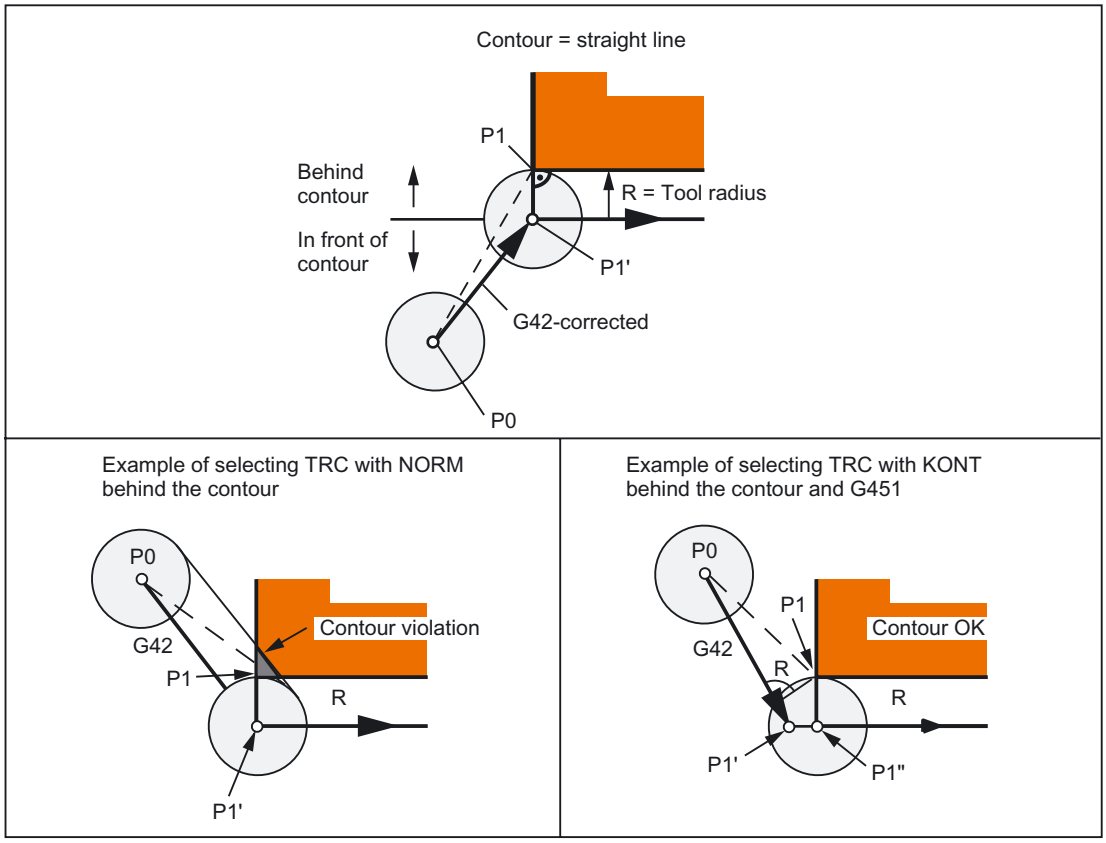


Figure 17-12 Example for selecting TRC with KONT or NORM in front of and behind the contour

- KONT and G450/G451 (corner behavior at outer corners) has a general effect and determines the approach and retraction behavior with TRC.
- When tool radius compensation is deselected, the retraction behavior is determined by the NORM/KONT instructions.

Supplementary conditions

The approach and retraction blocks are polynomials in the following two variants. Therefore, they are only available for control variants, which support polynomial interpolation.

- KONTT

  With KONTT, approach and retraction to/from the contour is with a constant tangent. The curvature at the block transition is not usually constant.

- KONTC

  With KONTC, not only the tangent but also the curvature is constant at the transition, with the result that **a jump in acceleration** can no longer occur on activation/deactivation.

Although KONTC includes the KONTT property, the constant tangent version KONTT is available on its own, because the constant curvature required by KONTC can produce undesired contours.

Axes

The continuity condition is observed in all **three** axes. It is thus possible to program a simultaneous path component perpendicular to the compensation plane for approach/retraction.

Only **linear blocks** are permitted for the original approach and retraction blocks with KONTT/KONTC. These programmed linear blocks are replaced in the control by the corresponding polynomial curves.

Exception

KONTT and KONTC are not available in 3D variants of tool radius compensation (CUT3DC, CUT3DCC, CUT3DF).

If they are programmed, the control switches internally to NORM without an error message.

Example for KONTC

The two figures below show a typical application for approach and retraction with constant curvature:

The full circle is approached beginning at the circle center point. The direction and curvature radius of the approach circle at the block end point are identical to the values of the next circle. Infeed takes place in the Z direction in both approach/retraction blocks simultaneously.

The associated NC program segment is as follows:

```
$TC_DP1[1,1]=121                                ; Milling tool
$TC_DP6 [1,1]=10                                ; Radius 10 mm
N10 G1 X0 Y0 Z60 G64 T1 D1 F10000
N20 G41 KONTC X70 Y0 Z0
N30 G2 I-70                                     ; Full circle
N40 G40 G1 X0 Y0 Z60
N50 M30
```

Explanation:

In this example, a full circle with a radius of 70 mm is machined in the X/Y plane. Since the tool has a radius of 10 mm, the resulting tool center point path describes a circle with a radius of 60 mm. The start/end points are at X0 Y0 Z60, with the result that a movement takes place in the Z direction at the same time as the approach/retraction movement in the compensation plane.

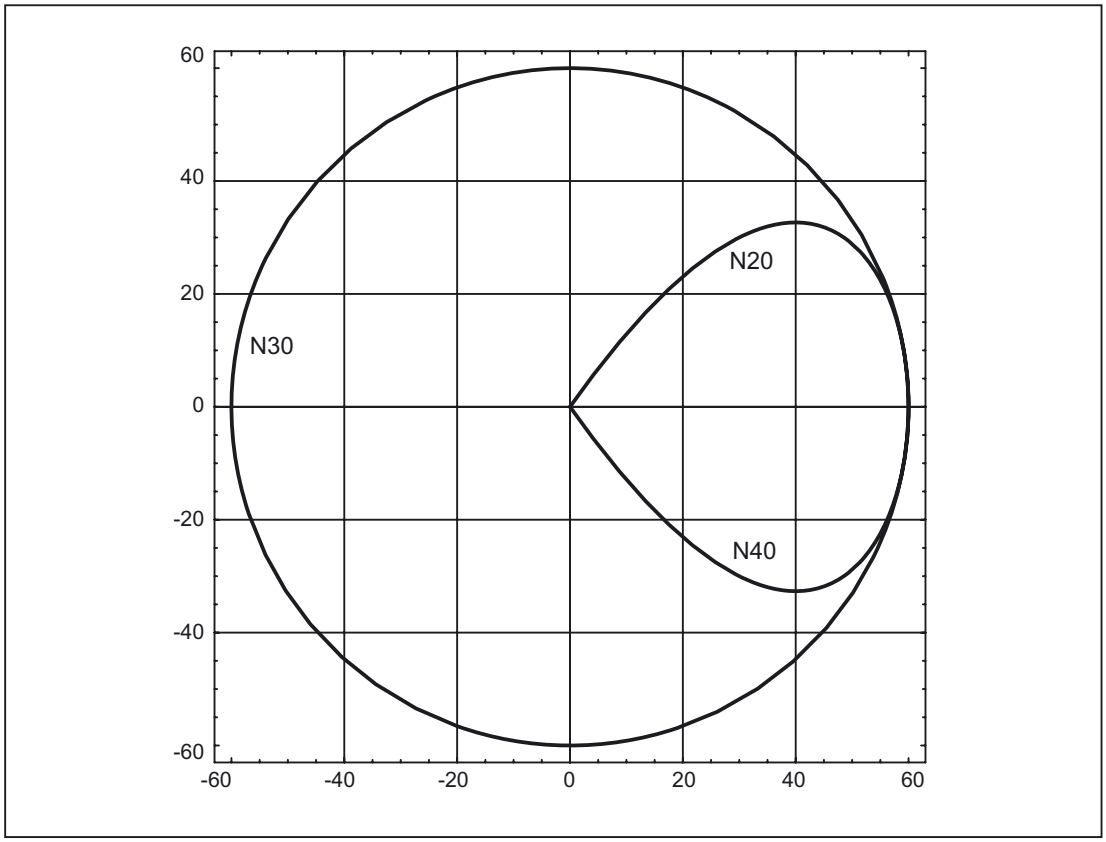


Figure 17-13 Approach and retraction with constant curvature during inside machining of a full circle: Projection in the X-Y plane.

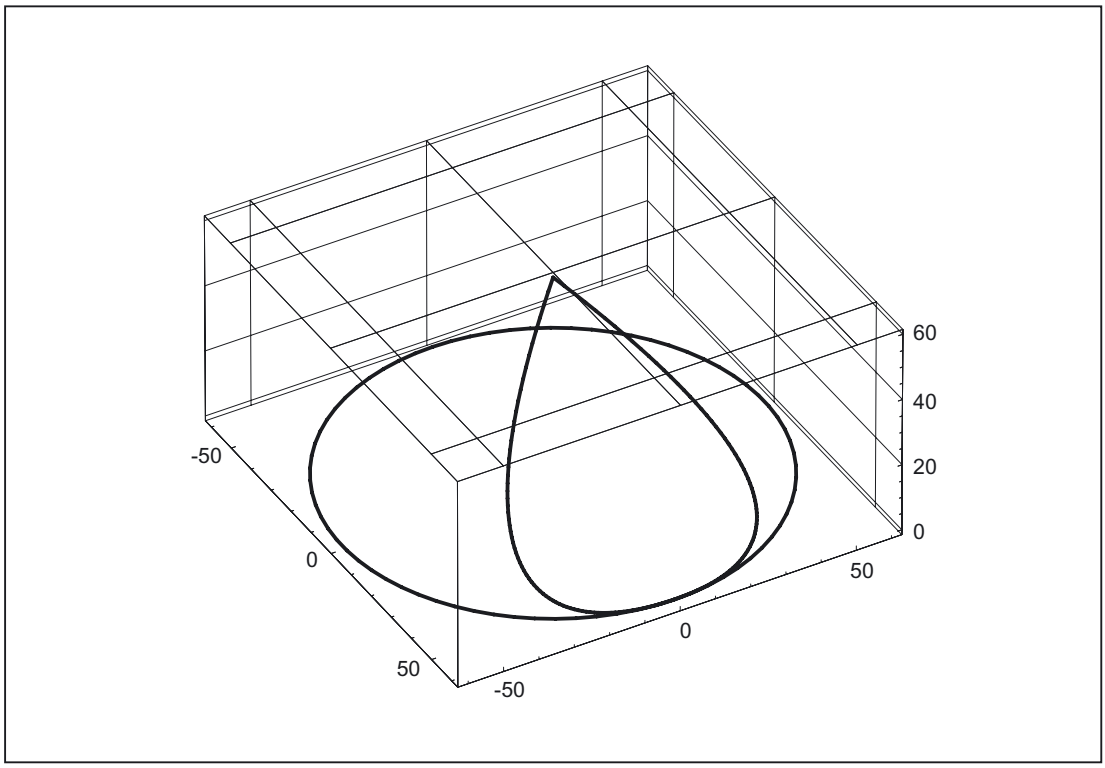


Figure 17-14 Approach and retraction with constant curvature during inside machining of a full circle: 3D representation.

KONTT and KONTC compared

The figure below shows the differences in approach/retraction behavior between KONTT and KONTC. A circle with a radius of 20 mm about the center point at X0 Y-40 is compensated with a tool with an external radius of 20 mm. The tool center point therefore moves along a circular path with radius 40 mm. The end point of the approach blocks is at X40 Y30. The transition between the circular block and the retraction block is at the zero point. Due to the extended continuity of curvature associated with KONTC, the retraction block first executes a movement with a negative Y component. This will often be undesired. This response does not occur with the KONTT retraction block. However, with this block, an acceleration step change occurs at the block transition.

If the KONTT or KONTC block is the approach block rather than the retraction block, the contour is exactly the same, but is simply machined in the opposite direction, i.e. the **approach and retraction behavior are symmetrical.**

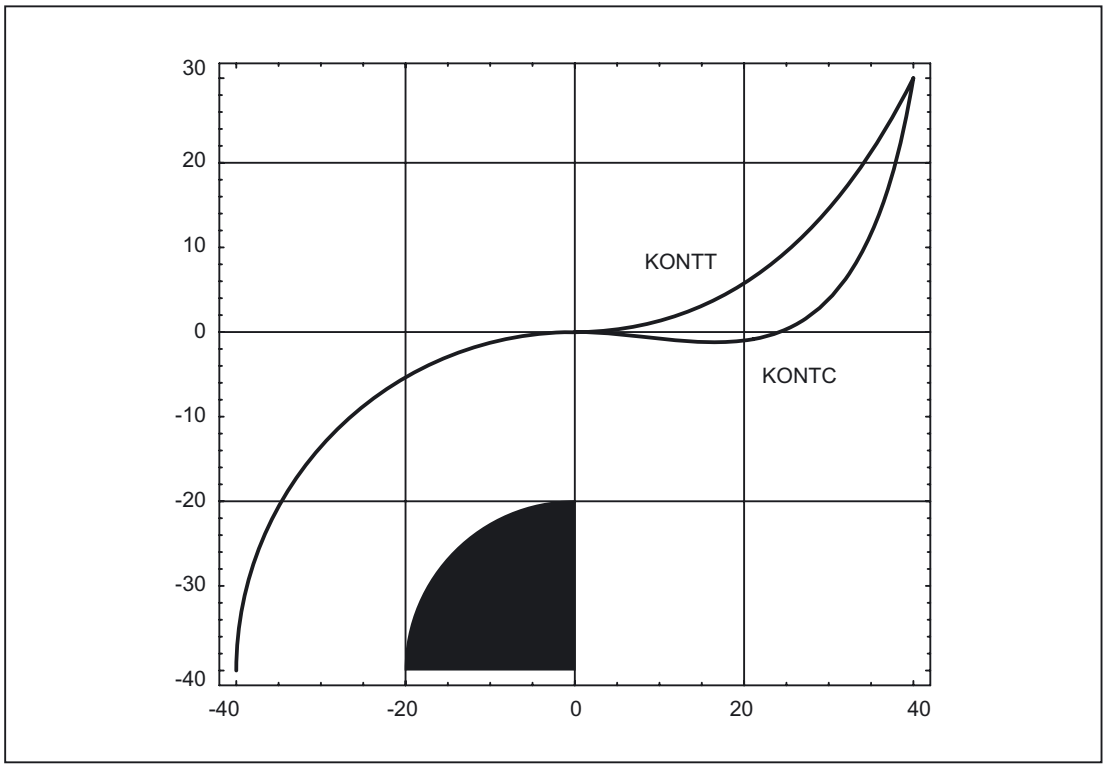


Figure 17-15 Differences between KONTT and KONTC

Note

The figure shows that a straight line bordering on the contour quadrant, e.g. to X20 Y-20, would be violated with KONTC on retraction/approach to X0, Y0.

17.5.4 Smooth approach and retraction

17.5.4.1 Function

Description

The SAR (Smooth Approach and Retraction) function is used to achieve a tangential approach to the start point of a contour, regardless of the position of the start point.

The approach behavior can be varied and adapted to special needs using a range of additional parameters.

The two functions, smooth approach and smooth retraction, are largely symmetrical. The following section is, therefore, restricted to a detailed description of approach; special reference is made to differences affecting retraction.

Sub-movements

There are maximum 4 sub-movements in case of soft retraction and approach with the following positions:

- Start point of the movement P_0
- Intermediate points P_1, P_2 and P_3
- End point P_4

Points P_0, P_3 and P_4 are always defined. Intermediate points P_1 and P_2 can be omitted, according to the parameters defined and the geometrical conditions.

On retraction, the points are traversed in the reverse direction, i.e. starting at P_4 and ending at P_0.

17.5.4.2 Parameters

The response of the smooth approach and retraction function is determined by up to 9 parameters:

Non-modal G code for defining the approach and retraction contour

This G code cannot be omitted.

- G147: Approach with a straight line
- G148: Retraction with a straight line
- G247: Approach with a quadrant
- G248: Retraction with a quadrant
- G347: Approach with a semicircle
- G348: Retraction with a semicircle

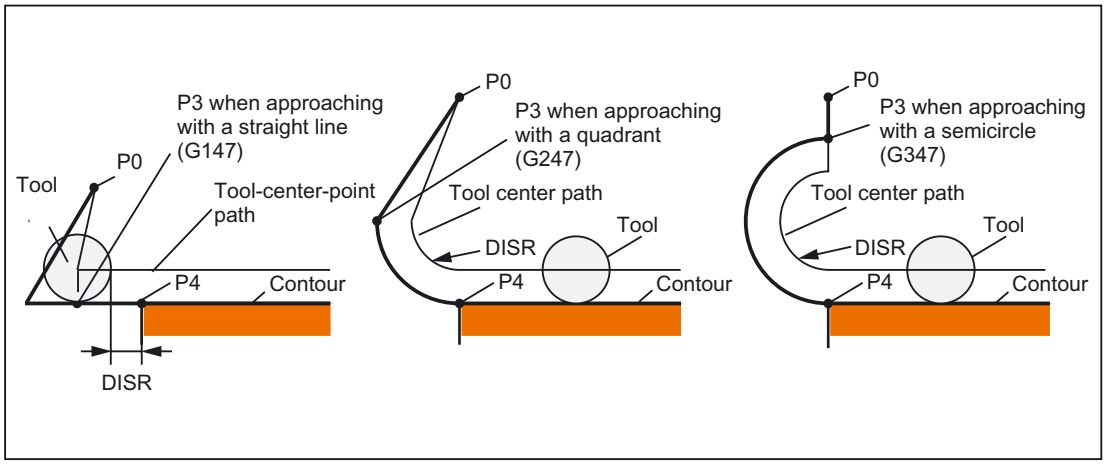


Figure 17-16 Approach behavior depending on G147 to G347 and DISR (with simultaneous activation of tool radius compensation)

Modal G code for defining the approach and retraction contour

This G code is only relevant if the approach contour is a quadrant or semicircle. The approach and retraction direction can be determined as follows:

- G140:

  Defining the approach and retraction direction using active tool radius compensation. ((G140 is the initial setting.)

  With positive tool radius:

  – G41 active → approach from left

  – G42 active → approach from right

  If no tool radius compensation is active (G40), the response is identical to G143. In this case, an alarm is not output. If the radius of the active tool is 0, the approach and retraction side is determined as if the tool radius were positive.

- G141:

  Approach contour from left, or retract to the left.

- G142:

  Approach contour from right, or retract to the right.

- G143:

  Automatic determination of the approach direction, i.e., the contour is approached from the side where the start point is located, relative to the tangent at the start point of the following block (P_4).

Note

The tangent at the end point of the preceding block is used accordingly on **retraction**. If the end point is not programmed explicitly on retraction, i.e., if it is to be determined implicitly, G143 is not permitted on retraction, since there is a mutual dependency between the approach side and the position of the end point. If G143 is programmed in this case, an alarm is output. The same applies if, when G140 is active, an automatic switchover to G143 takes place as a result of an inactive tool radius compensation.

Modal G code (G340, G341), which defines the subdivision of the movement into individual blocks from the start point to the end point

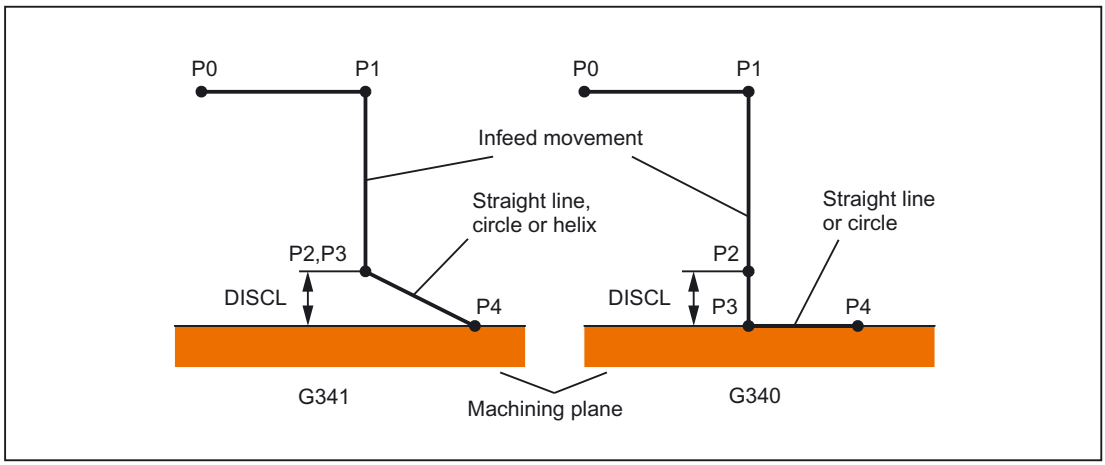


Figure 17-17 Sequence of the approach movement depending on G340/G341

G340: The approach characteristic from P_0 to P_4 is shown in the figure.

If G247 or G347 is active (quadrant or semicircle) and start point P_3 is outside the machining plane defined by the end point P_4, a helix is inserted instead of a circle. Point P_2 is not defined or coincides with P_3.

The circle plane or the helix axis is determined by the plane, which is active in the SAR block (G17 - G19), i.e., the projection of the start tangent is used by the following block, instead of the tangent itself, to define the circle.

The movement from point P_0 to point P_3 takes place along two straight lines at the velocity valid before the SAR block.

G341: The approach characteristic from P_0 to P_4 is shown in the figure.

P_3 and P_4 are located within the machining plane, with the result that a circle is always inserted instead of a helix with G247 or G347.

Note

Active, rotating frames are included in all cases where the position of the active plane G17 - G19 (circle plane, helix axis, infeed movements perpendicular to the active plane) is relevant.

DISR

DISRSpecifies the length of a straight approach line or the radius of an approach arc.

Retraction/approach with straight lines

On approach/retraction along a straight line, DISR specifies the distance from the cutter edge to the start point of the contour, i.e., the length of the straight line with active TRC is calculated as the total of the tool radius and the programmed value of DISR.

An alarm is displayed:

- If DISR is negative and the amount is greater than the tool radius (the length of the resulting approach line is less than or equal to zero).

Retraction/approach with circles

Approach/retraction with circles DISR indicates always the radius of the tool center point path. If tool radius compensation is activated, a circle is generated internally, the radius of which is dimensioned such that the tool center path is derived, in this case also, from the programmed radius.

An alarm is output on approach and retraction with circles:

- If the radius of the circle generated internally is zero or negative
- If DISR is not programmed
- If the radius value ≤ 0.

DISCL

DISCLspecifies the distance from point P2 from the machining plane.

If the position of point P_2 is to be specified by an absolute reference on the axis perpendicular to the circle plane, the value must be programmed in the form DISCL = AC(....).

If DISCL is not programmed, points P_1, P_2 and P_3 are identical with G340 and the approach contour is mapped from P_1 to P_4.

The system checks that the point defined by DISCL lies between P_1 and P_3, i.e., in all movements, which have a component perpendicular to the machining plane (e.g., infeed movements, approach movements from P_3 to P_4), this component must have the same leading sign. It is not permitted to change direction. An alarm is output if this condition is violated.

On detection of a direction reversal, a tolerance is permitted that is defined by the machine data:

MD20204 $MC_WAB_CLEARANCE_TOLERANCE (direction reversal on SAR).

However, if P_2 is outside the range defined by P_1 and P_3 and the deviation is less than or equal to this tolerance, it is assumed that P_2 is in the plane defined by P_1 and/or P_3.

Example:

An approach is made with G17 starting at position Z=20 of point P_1. The SAR plane defined by P_3 is at Z=0. The point defined by DISCL must, therefore, lie between these two points. MD20204=0.010. If P_2 is between 20.000 and 20.010 or between 0 and -0.010, it is assumed that the value 20.0 or 0.0 is programmed. The alarm is output if the Z position of P_2 is greater than 20.010 or less than -0.010.

Depending on the relative position of start point P_0 and end point P_4 with reference to the machining plane, the infeed movements are performed in the negative (normal for approach) or positive (normal for retraction) direction, i.e., with G17 it is possible for the Z component of end point P_4 to be greater than that of start point P_0.

Programming the end point P_4 (or P_0 for retraction) generally with X... Y... Z...

Possible ways of programming the end point $P_4$for approach

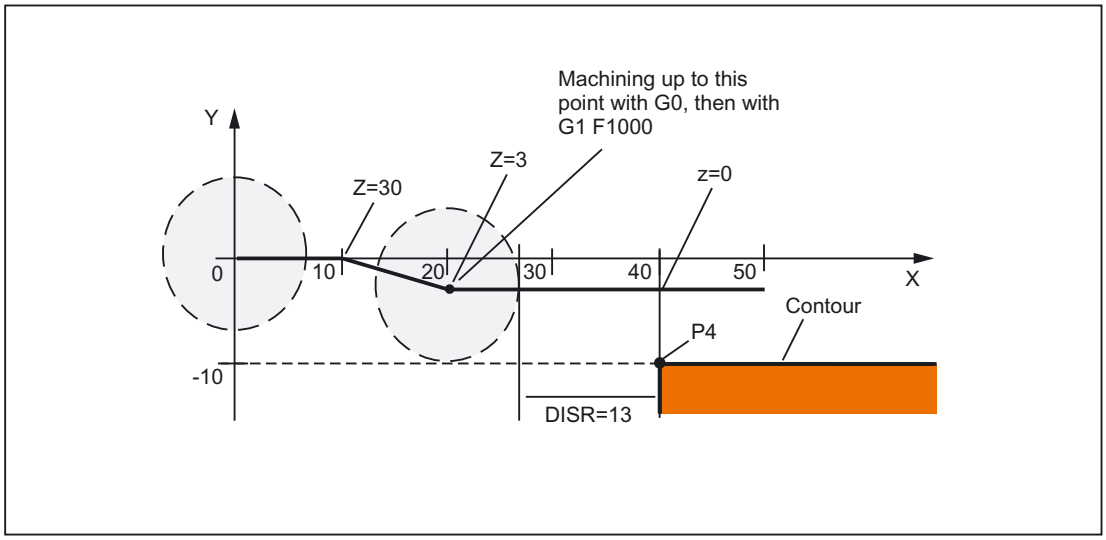
End point P_4 can be programmed in the actual SAR block.

P_4 can be determined by the end point of the next traversing block.

Further blocks (dummy blocks) can be inserted between the SAR block and the next traversing block without moving the geometry axes.

The end point is deemed to have been programmed in the actual SAR block for approach if at least one geometry axis is programmed on the machining plane (X or Y with G17). If only the position of the axis perpendicular to the machining plane (Z with G17) is programmed in the SAR block, this component is taken from the SAR block, but the position in the plane is taken from the following block. In this case, an alarm is output if the axis perpendicular to the machining plane is also programmed in the following block.

Example:




```
$TC_DP1[1,1]=120                                    ; Milling tool T1/D1
$TC_DP6[1,1]=7                                      ; Tool with 7 mm radius

N10 G90 G0 X0 Y0 Z30 D1 T1
N20 X10
N30 G41 G147 DISCL=3 DISR=13 Z=0 F1000
N40 G1 X40 Y-10
N50 G1 X50
...
...
```

N30/N40 can be replaced by:

```
N30 G41 G147 DISCL=3 DISR=13 X40 Y-10 Z0 F1000
```

or:

```
N30 G41 G147 DISCL=3 DISR=13 F1000
N40 G1 X40 Y-10 Z0
```

Possible ways of programming the end point P_0 for retraction

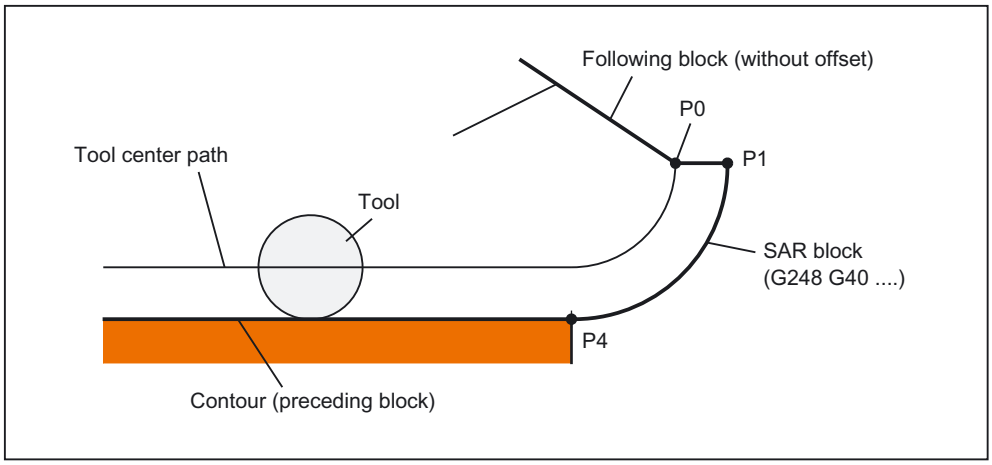
The end position is always taken from the SAR block, no matter how many axes have been programmed. We distinguish between the following situations:

1. No geometry axis is programmed in the SAR block.

   In this case, the contour ends at point P_2 (or at point P_1, if P_1 and P_2 coincide). The position in the axes, which describe the machining plane, is determined by the retraction contour (end point of the straight line or arc). The axis component perpendicular to this is defined by `DISCL`. If, in this case, `DISCL = 0`, the movement takes place completely within the plane.

2. Only the axis perpendicular to the machining plane is programmed in the SAR block.

   In this case, the contour ends at point P_1. The position of the two other axes is determined in the same way as in 1.



   Retraction with SAR with simultaneous deactivation of TRC

   If the SAR retraction block is also used to deactivate tool radius compensation, in the case of 1. and 2., an additional path from P_1 to P_0 is inserted such that no movement is produced when tool radius compensation is deactivated at the end of the retraction contour, i.e., this point defines the tool center point and not a position on a contour to be corrected.

3. At least one axis of the machining plane is programmed.

   The second axis of the machining plane can be determined modally from its last position in the preceding block. The position of the axis perpendicular to the machining plane is generated as described in 1. or 2., depending on whether this axis is programmed or not. The position generated in this way defines the end point P_0.

   No special measures are required for deselection of tool radius compensation, because the programmed point P_0 already directly defines the position of the tool center point at the end of the complete contour.

   The start and end points of the SAR contour (P_0 and P_4) can coincide on approach and retraction.

Velocity of the preceding block (typically G0).

All movements from point P_0 to point P_2 are performed at this velocity, i.e., the movement parallel to the machining plane and the part of the infeed movement up to the safety clearance.

Programming the feedrate with FAD

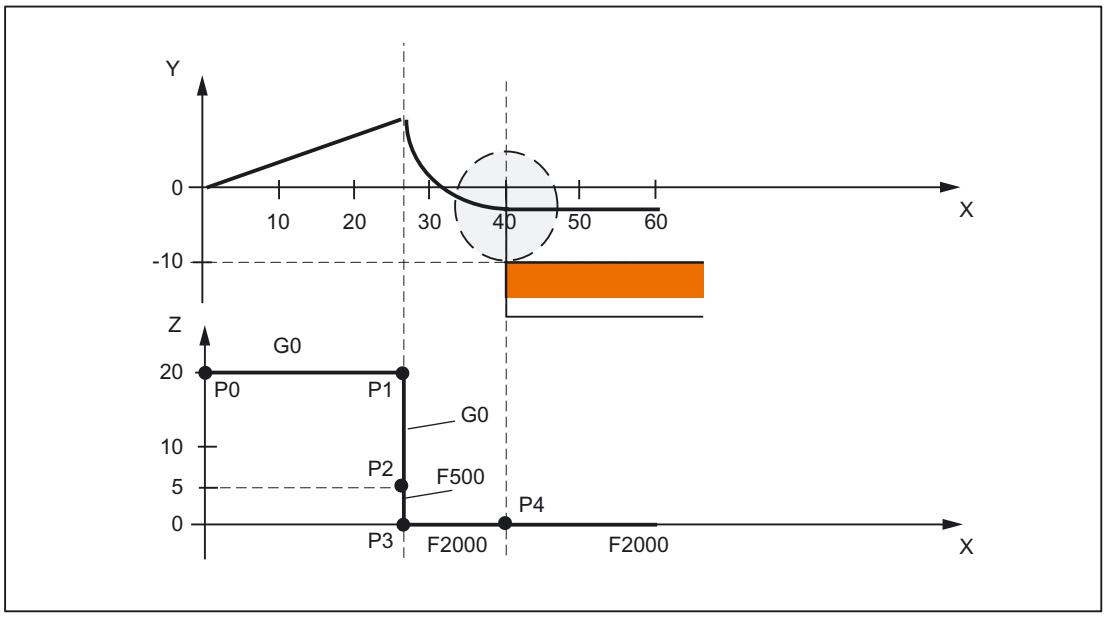
FAD programmed with ...	
G340	Feedrate from P_2 or P_3 to P_4.
G341	Feedrate of the infeed movement perpendicular to the machining plane from P_2 to P_3.

If FAD is not programmed, this part of the contour is traversed at the velocity, which is active modally from the preceding block, in the event that no F command defining the velocity is programmed in the SAR block.

Programmed response:

FAD=0 or negative	→	Alarm Output
FAD=...	→	Programmed value acts in accordance with the active G code of group 15 (feed type; G93, G94, etc.)
FAD=PM(...)	→	Programmed value is interpreted as linear feed (like G94), irrespective of the active G code of group 15
FAD=PR(...)	→	Programmed value is interpreted as revolutional feed (like G95), irrespective of the active G code of group 15

Example:




```
$TC_DP1[1,1]=120                                    ;Milling tool T1/D1
$TC_DP6[1,1]=7                                      ;Tool with 7mm radius

N10 G90 G0 X0 Y0 Z20 D1 T1
N20 G41 G341 G247 DISCL=AC(5) DISR=13FAD 500 X40 Y-10 Z=0 F2000
N30 X50
N40 X60
...
```

Programming feed F

This feed value is effective from point P_3 (or from point P_2, if FAD is not programmed). If no F command is programmed in the SAR block, the speed of the preceding block is valid. The velocity defined by FAD is not used for following blocks.

17.5.4.3 Velocities

Velocities at approach

In both approach diagrams below, it is assumed that no new velocity is programmed in the block following the SAR block. If this is not the case, the new velocity comes into effect after point P_4.

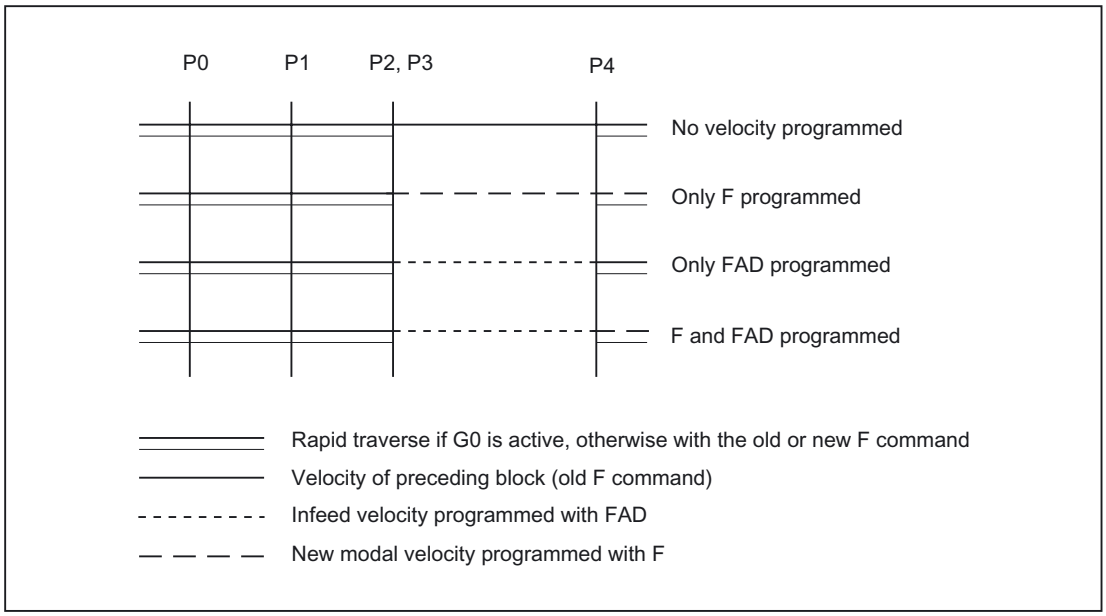


Figure 17-18 Velocities in the SAR subblocks on approach with G340

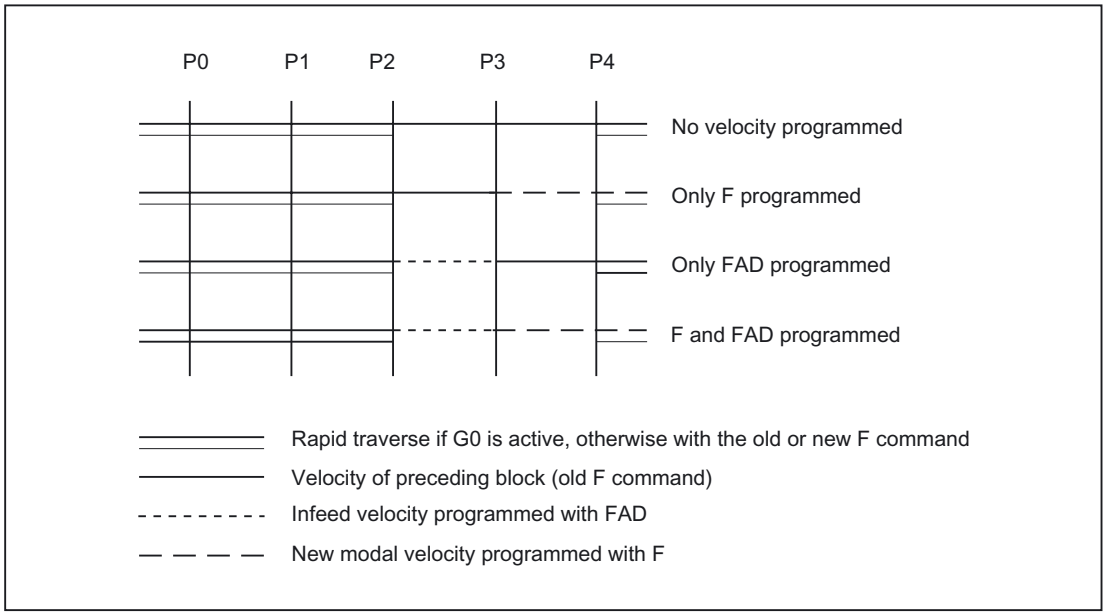


Figure 17-19 Velocities in the SAR subblocks on approach with G341

Velocities at retraction

During retraction, the rolls of the modally active feedrate from the previous block and the programmed feedrate value in the SAR block are interchanged, i.e., the actual retraction contour (straight line, circle, helix) is traversed with the old feedrate value and a new velocity programmed with the F word applies from point P_2 up to P_0.

If even retraction is active and `FAD` is programmed, the path from P_3 to P_2 is traversed with `FAD`, otherwise it is traversed with the old velocity. The last F command programmed in a preceding block always applies for the path from P_4 to P_2. `G0` has no effect in these blocks.

Traversing from P_2 to P_0 takes place with the F command programmed in the SAR block or, if no F command is programmed, with the modal F command from a preceding block. This applies on the condition that `G0` is not active.

If rapid traverse is to be used on retraction in the blocks from P_2 to P_0, `G0` must be activated before the SAR block or in the SAR block itself. If an additional F command is programmed in the actual SAR blocks, it is then ineffective. However, it remains modally active for following blocks.

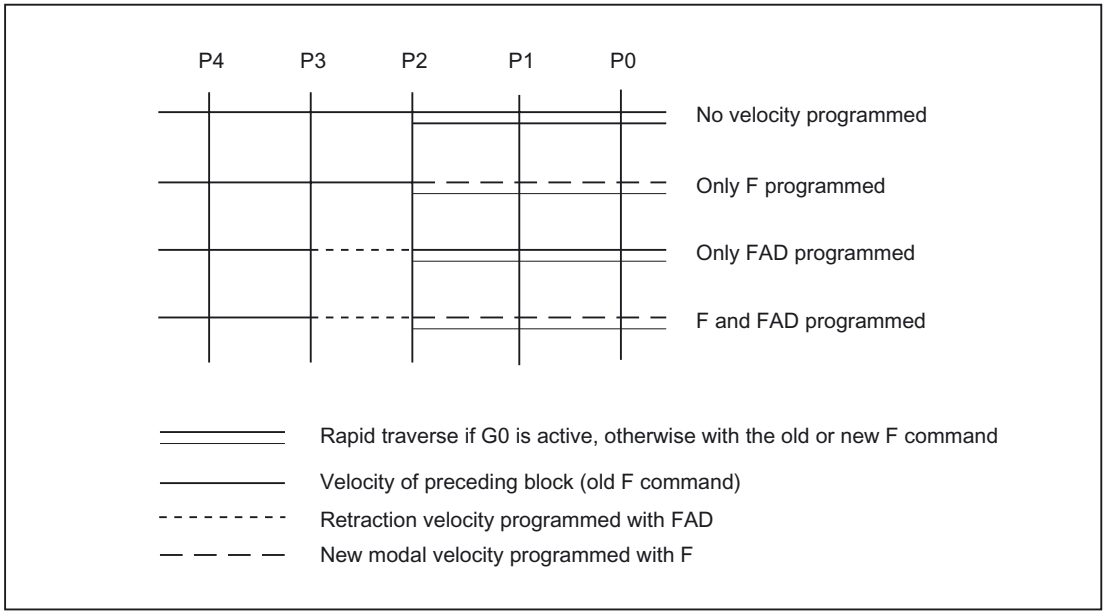


Figure 17-20 Velocities in the SAR subblocks on retraction

17.5.4.4 System variables

Points P_3 and P_4 can be read in the WCS as system variables during approach.

$P_APR:	Read P_3 (start point) in WCS	
$P_AEP:	Read P_4 (contour start point) in WCS	
$P_APDV	=1	If the content of $P_APR and $P_AEP is valid, i.e., if these contain the position values belonging to the last SAR approach block programmed.
	=0	The positions of an older SAR approach block are read.

Changing the WCS between the SAR block and the read operation has no effect on the position values.

17.5.4.5 Supplementary conditions

Supplementary conditions

- Any further NC commands (e.g. auxiliary function outputs, synchronous axis movements, positioning axis movements, etc.) can be programmed in an SAR block.

  These are executed in the first subblock on approach and in the last subblock on retraction.

- If the end point P_4 is not taken from the SAR block but from a subsequent traversing block, the actual SAR contour (straight line, quadrant or semicircle) is traversed in this block.

  The last subblock of the original SAR block does not then contain traversing information for geometry axes. It is always output, however, because further actions (e.g. single axes) may have to be executed in this block.

- At least two blocks must always be taken into consideration:
  - The SAR block itself
  - The block, which defines the approach or retraction direction

  Further blocks can be programmed between these two blocks.

  The number of possible dummy blocks is limited with the machine data:

  MD20202 $MC_WAB_MAXNUM_DUMMY_BLOCKS (maximum number of blocks with no traversing motions with SAR).

- If tool radius compensation is activated simultaneously in an approach block the first linear block of the SAR contour is the block in which activation takes place.

  The complete contour generated by the SAR function is treated by tool radius compensation as if it has been programmed explicitly (collision detection, calculation of intersection, approach behavior NORM/KONT).

- The direction of the infeed motion and the position of the circle plane or the helix axis are defined exclusively by the active plane (G17 - G19) - rotated with an active frame where appropriate.

- On approach, a preprocessor stop must not be inserted between the SAR block and the following block which defines the direction of the tangent.

  Whether programmed explicitly or inserted automatically by the control, a preprocessor stop results, in this case, in an alarm.

Behavior with REPOS

If an SAR cycle is interrupted and repositioned, it resumes at the point of interruption on RMIBL. With RMEBL, the contact point is the end point of the last SAR block; with RMBBL, it is the start point of the first SAR block.

If RMIBL is programmed together with DISPR (reapproach at distance DISPR in front of interruption point), the reapproach point can appear in a subblock of the SAR cycle before the interruption subblock.

17.5.4.6 Examples

Example 1

The following conditions must be true:

- Smooth approach is activated in block N20
- X=40 (end point); Y=0; Z=0
- Approach movement performed with quadrant (G247)
- Approach direction not programmed, G140 is valid, i.e. because TRC is active (G42) and compensation value is positive (10), the contour is approached from the right
- Approach circle generated internally (SAR contour) has radius 20, so that the radius of the tool center path is equal to the programmed value DISR=10
- Because of G341, the approach movement takes place with a circle in the plane, resulting in a start point at (20, -20, 0)
- Because DISCL=5, point P2 is at position (20, -20, 5) and, because of Z30, point P1 is in N10 at (20, -20, 30)

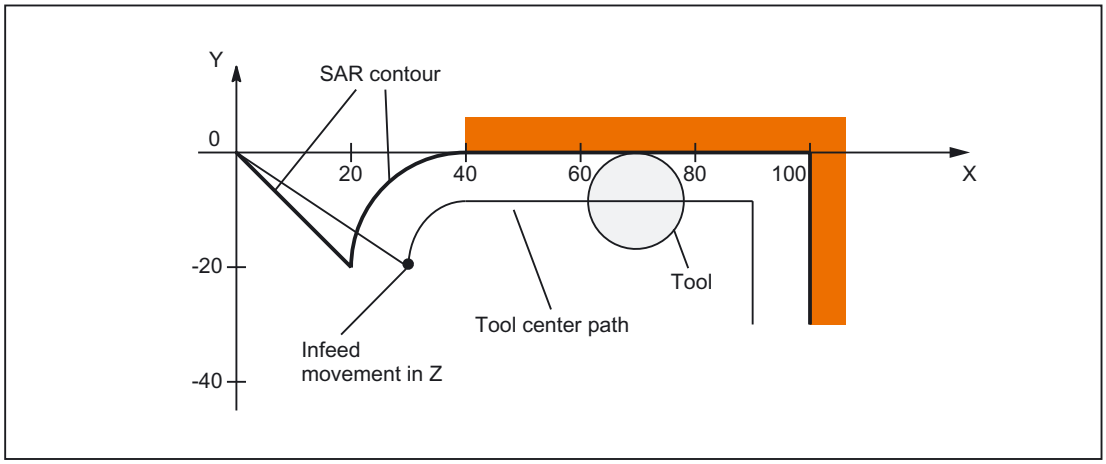


Figure 17-21 Contour example 1

Part program:

Program code	Comment
$TC_DP1[1,1]=120	; Tool definition T1/D1
$TC_DP6[1,1]=10	; Radius
N10 G0 X0 Y0 Z30	
N20 G247 G341 G42 NORM D1 T1 Z0 FAD=1000 F=2000 DISCL=5 DISR=10	
N30 X40	
N40 X100	
N50 Y-30	
...	

Example 2

The following conditions must be true for approach:

- Smooth approach is activated in block N20
- Approach movement performed with quadrant (G247)
- Approach direction not programmed, G140 is valid, i.e. because TRC is active (G41), the contour is approached from the left
- Contour offset OFFN=5 (N10)
- Current tool radius=10, and so the effective compensation radius for TRC=15; the radius of the SAR contour is thus equal to 25, with the result that the radius of the tool center path is equal to DISR=10
- The end point of the circle is obtained from N30, since only the Z position is programmed in N20
- Infeed motion
  - From Z20 to Z7 (DISCL=AC(7)) with rapid traverse
  - Then on to Z0 with FAD=200
  - Approach circle in X-Y-plane and following blocks with F1500

    (In order that this velocity becomes effective in the following blocks, the active G-code G0 in N30 must be overwritten with G1. Otherwise, the contour would continue to be machined with G0.)

The following conditions must be true for retraction:

- Smooth retraction is activated in block N60
- Retraction movement performed with quadrant (G248) and helix (G340)
- FAD not programmed, since irrelevant for G340
- Z=2 in the start point; Z=8 in the end point, since DISCL=6
- When DISR=5, the radius of SAR contour=20; that of the tool center point path=5
- After the circle block, the retraction movement leads from Z8 to Z20 and the movement is parallel to the XY plane up to the end point at X70 Y0

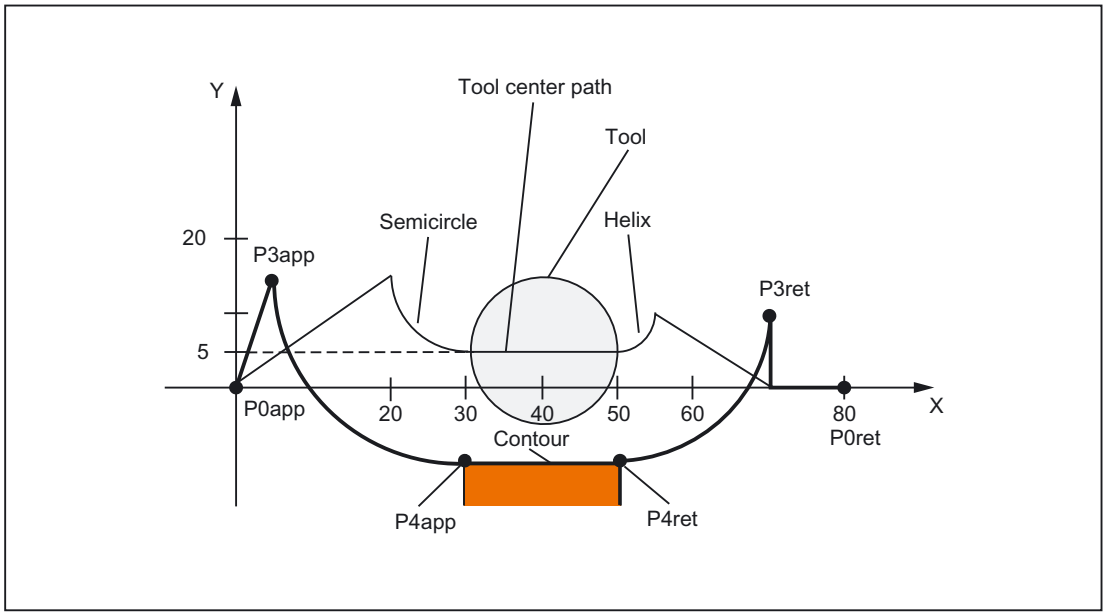


Figure 17-22 Contour example 2

Part program:

Program code	Comment
$TC_DP1[1,1]=120	; Tool definition T1/D1
$TC_DP6[1,1]=10	; Radius
N10 G0 X0 Y0 Z20 G64 D1 T1 OFFN = 5	; (P0app)
N20 G41 G247 G341 Z0 DISCL = AC(7) DISR = 10 F1500 FAD=200	; (P3app)
N30 G1 X30 Y-10	; (P4app)
N40 X40 Z2	
N50 X50	; (P4ret)
N60 G248 G340 X70 Y0 Z20 DISCL = 6 DISR = 5 G40 F10000	; (P3ret)
N70 X80 Y0	; (P0ret)
N80 M 30	

Note

The contour generated in this way is modified by tool radius compensation, which is activated in the SAR approach block and deactivated in the SAR retraction block.

The tool radius compensation allows for an effective radius of 15, which is the sum of the tool radius (10) and the contour offset (5). The resulting radius of the tool center path in the approach block is therefore 10, and 5 in the retraction block.

17.5.5 Deselecting the TRC (G40)

G40 instruction

TRC is deselected with the G40 instruction.

Special points to be noted

- TRC can only be deselected in a program block with G0 (rapid traverse) or G1 (linear interpolation).
- If D0 is programmed when tool radius compensation is active, compensation is not deselected and error message 10750 is output.
- If a geometry axis is programmed in the block with the tool radius compensation deselection, then the compensation is deselected even if it is not on the current plane.

17.5.6 Compensation at outside corners

G450/G451

The G functions G450/G451 can be used to control the response with discontinuous block transitions at outside corners:

Command	Meaning
G450	Discontinuous block transitions with transition circle
G451	Discontinuous block transitions with intersection of equidistant paths

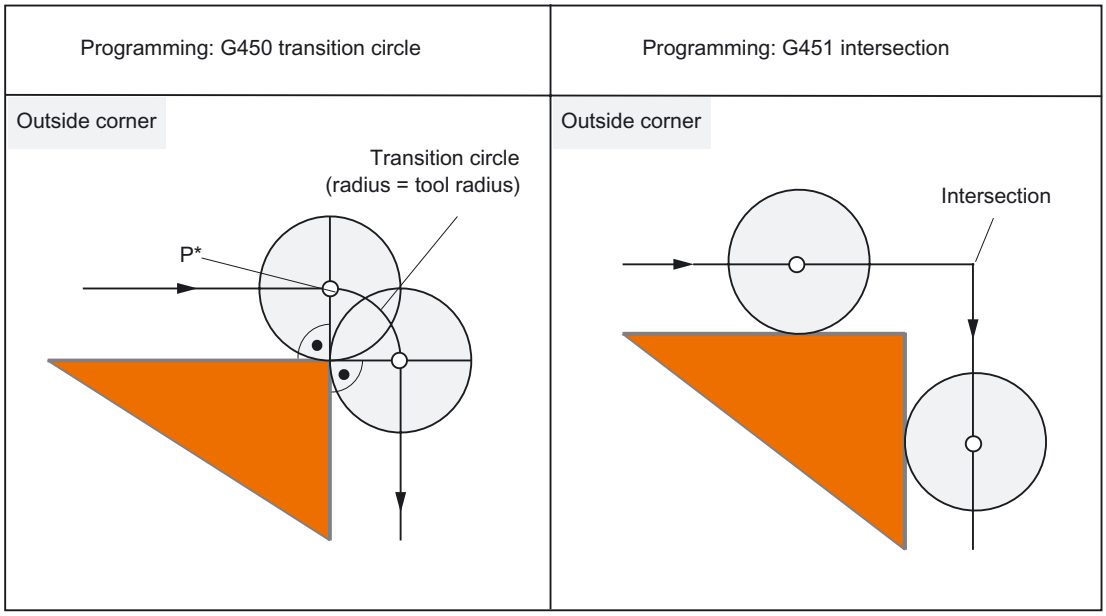


Figure 17-23 Example of a 90 degree outside corner with G450 and G451

G450 (transition circle)

With the G function G450 active, on outside corners, the center point of the tool travels a circular path along the tool radius. The circular path starts with the normal position (perpendicular to the path tangent) at the end point of the previous path section (program block) and ends in normal position at the start point of the new path section (program block).

Where outside corners are very flat, the response with G450 (transition circle) and G451 (intersection) becomes increasingly similar (see "Very flat outside corners").

If pointed outside corners are desired, the tool must be retracted from the contour (see Section "DISC").

DISC

The G450 transition circle does not produce sharp outside contour corners because the path of the tool center point through the transition circle is controlled so that the cutting edge stops at the outside corner (programmed position). When sharp outside corners are to be machined with G450, the DISC instruction can be used to program an overshoot. Thus, the transition circle becomes a conic and the tool cutting edge retracts from the outside corner.

The range of values of the DISC instruction is 0 to 100, in increments of 1.

Value	Meaning
DISC = 0	Overshoot disabled, transition circle active
DISC = 100	Overshoot large enough to theoretically produce a response similar to intersection (G451).

The programmable maximum value for DISC can be set via the machine data:
MD20220 $MC_CUTCOM_MAX_DISC (max. value for DISC).
Values greater than 50 are generally not advisable with DISC.

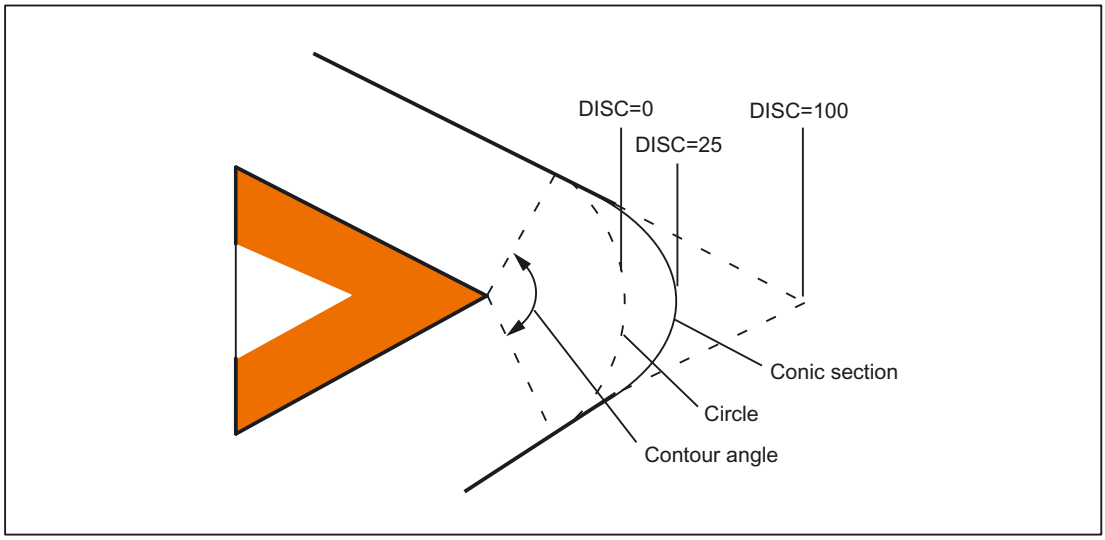


Figure 17-24 Example: Overshoot with DISC= 25

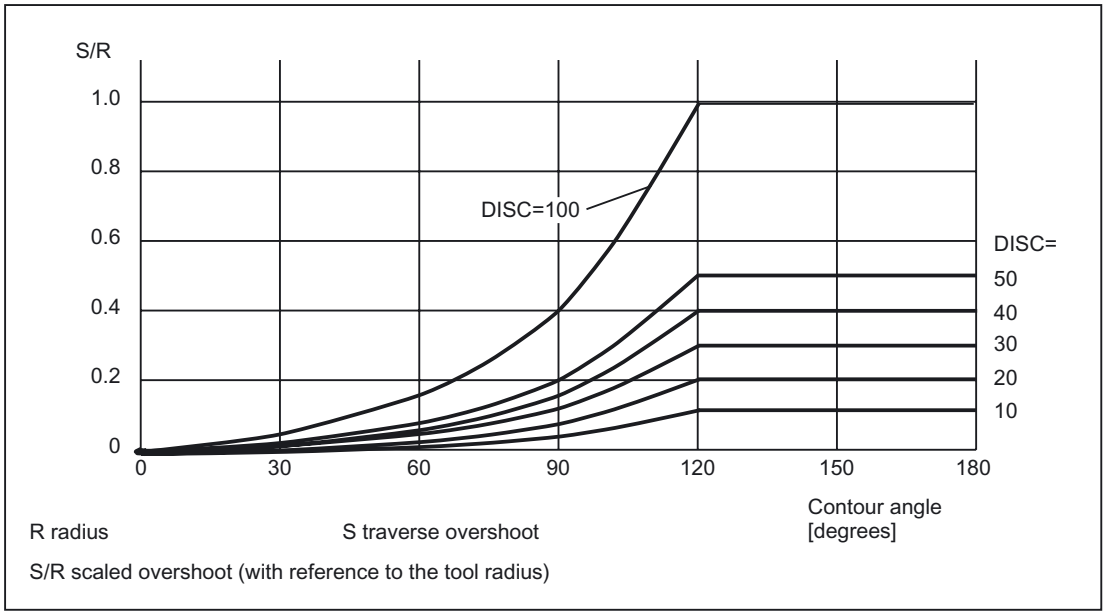


Figure 17-25 Overshoot with DISC depending on contour angle

G451 (intersection)

If G function G451 is active, the position (intersection) resulting from the path lines (straight line, circle or helix only) located at a distance of the tool radius to the programmed contour (center-point path of the tool), is approached. Spines and polynomials are never extended.

Very pointed outside corners

Where outside corners are very pointed, G451 can result in excessive idle paths. Therefore, the system switches automatically from G451 (intersection) to G450 (transition circle, with DISC where appropriate) when outside corners are very pointed.

The threshold angle (contour angle) for this automatic switchover (intersection point → transition circle) can be specified in the machine data:

MD20210 $MC_CUTCOM_CORNER_LIMIT (Max. angle for compensation blocks with tool radius compensation).

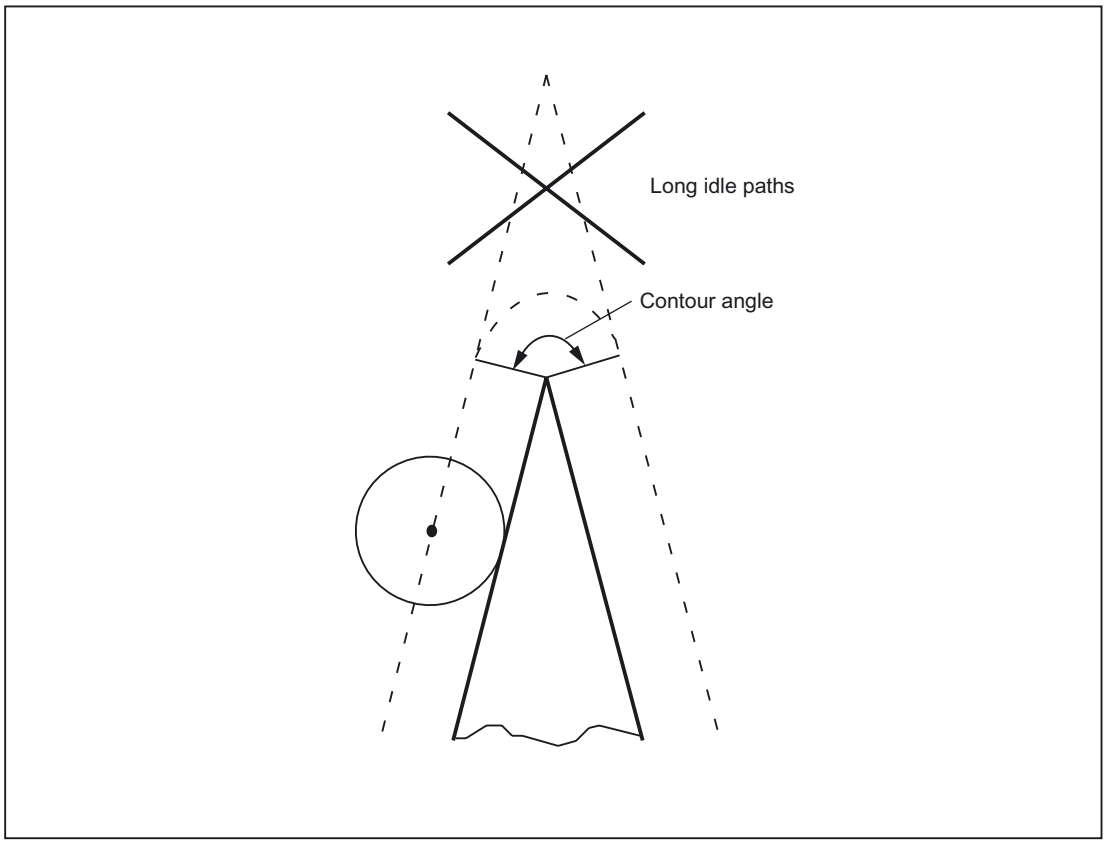


Figure 17-26 Example of automatic switchover to transition circle

Very flat outside corners

Where outside corners are very flat, the response with G450 (transition circle) and G451 (intersection) becomes increasingly similar. In this case, it is no longer advisable to insert a transition circle. One reason why it is not permitted to insert a transition circle at these outside corners with 5-axis machining is that this would impose restrictions on speed in contouring mode (G64). Therefore, the system switches automatically from G450 (transition circle, with DISC where appropriate) to G451 (intersection) when outside corners are very flat.

The threshold angle (contour angle) for this automatic switchover (transition circle → intersection point) can be specified in the machine data:

MD20230 $MC_CUTCOM_CURVE_INSERT_LIMIT (Max. angle for intersection calculation with tool radius compensation).

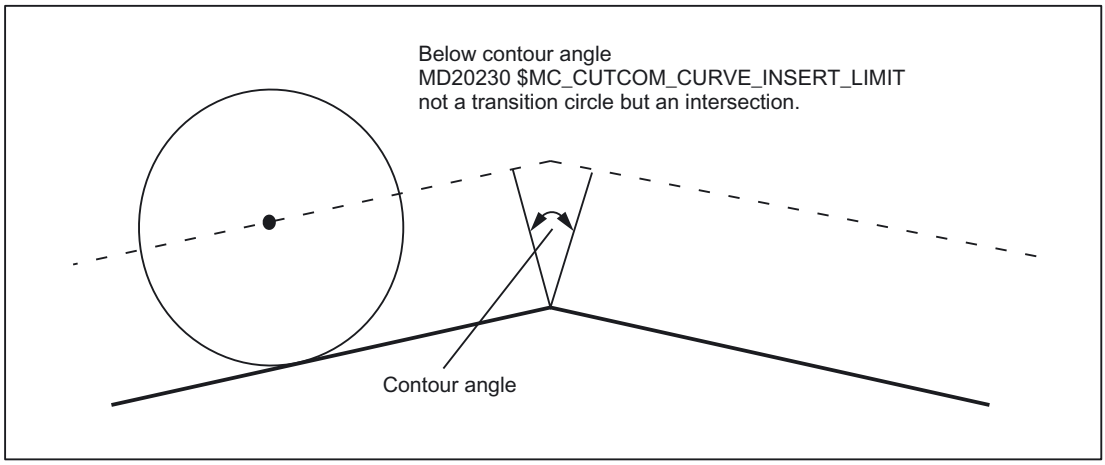


Figure 17-27 Example of automatic switchover to intersection

17.5.7 Compensation and inner corners

Intersection

If two consecutive blocks form an inside corner, an attempt is made to find a point at which the two equidistant paths intersect. If an intersection is found, the programmed contour is shortened to the intersection (first block shortened at end, second block shortened at beginning).

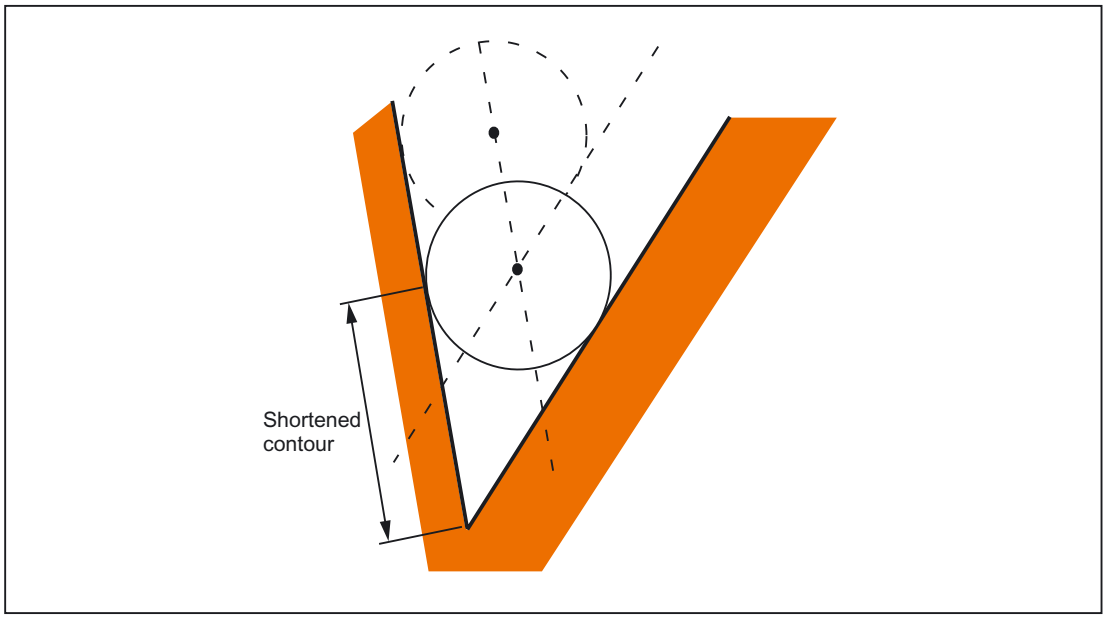


Figure 17-28 Example of a shortened contour

No intersection

In certain cases, no intersection is found between two consecutive blocks for inside corners (see figure below).

Predictive contour calculation

If no intersection is found between two consecutive blocks, the control automatically checks the next block and attempts to find an intersection with the equidistant paths of this block (see figure below: intersection S). This automatic check of the next block (predictive contour calculation) is always performed until a number of blocks defined via machine data has been reached.

MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS (blocks for predictive contour calculation for TRC).

If no intersection is found within the number of blocks defined for the check, program execution is interrupted and an alarm is output.

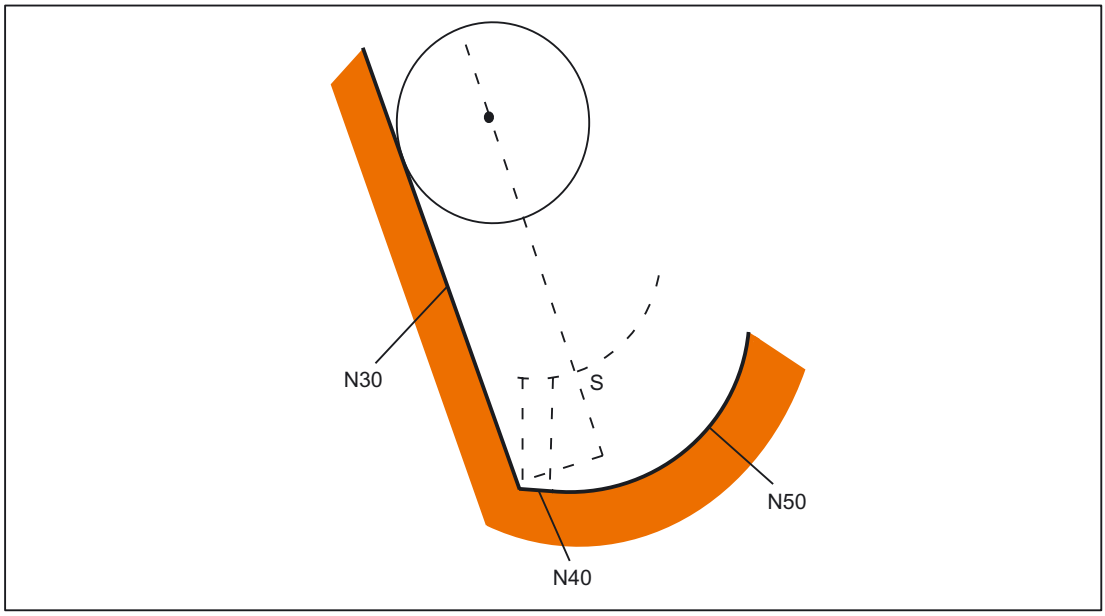


Figure 17-29 If there is no intersection between N30 and block N40, the intersection between block N30 and block N50 is calculated.

Multiple intersections

It can be the case with inside corners that predictive contour calculation finds multiple intersections of the equidistant paths in several consecutive blocks. In these cases, the last intersection is always used as the valid intersection:

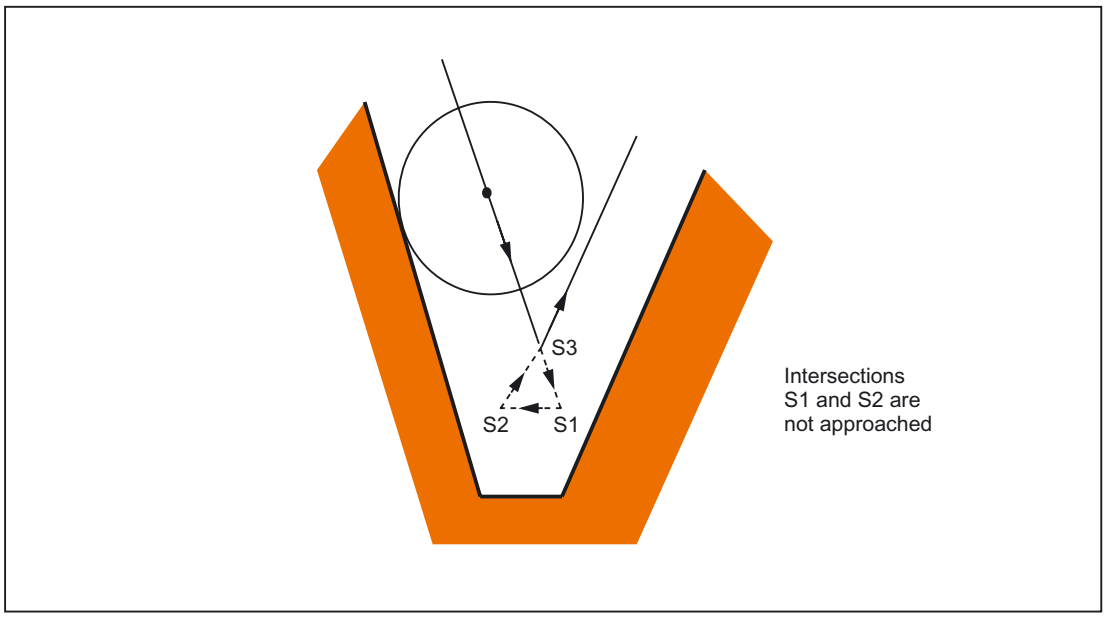


Figure 17-30 Example: Inside corner with TRC without contour violation (predicting 3 blocks)

For further information, see also Section "Collision detection and bottleneck detection (Page 1644)", "Collision monitoring".

Special features

Where multiple intersections with the next block are found, the intersection nearest the start of the next block applies.

17.5.8 Collision detection and bottleneck detection

Collision detection

Collision detection (bottleneck detection) examines whether the equidistant paths of non-consecutive blocks intersect. If an intersection is found, the response is the same as for inside corners with multiple intersections: The last intersection to be found is valid:

The maximum number of blocks used for the predictive check can be entered in the machine data:

MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS (blocks for predictive contour calculation for TRC).

Programming

Collision detection can be activated or deactivated in the program:

Command	Significance
CDON	Collision detection ON
CDOF	Collision detection OFF
CDOF2	Collision detection OFF

With CDOF, the search for an intersection initially examines two consecutive blocks. Other blocks are not included in the search. If an intersection is found between adjacent blocks, no further blocks are examined. With outside corners, an intersection can always be found between two consecutive blocks.

Predictive examination of more than two adjacent blocks is thus possible with CDON and CDOF.

Note

CDOF2 is only effective for 3D peripheral milling and has the same significance as CDOF for all other types of machining (e.g. 3D face milling).

Omission of block

If an intersection is detected between two blocks, which are not consecutive, none of the motions programmed between these blocks on the compensation plane are executed. All other motions and executable instructions (M commands, traversal of positioning axes, etc.) contained in the omitted blocks are executed at the intersection position in the sequence, in which they are programmed in the NC program.

Warning 10763

If a block has been omitted as a result of the collision or bottleneck detection functions, warning 10763 is output. The program is not interrupted.

This warning is suppressed if bit 1 is enabled in machine data:

MD11410 $MN_SUPPRES_ALARM_MASK (Mask for suppressing special alarm outputs).

Special points to be noted

When the intersections of non-consecutive blocks are checked, it is not the programmed original contours that are examined, but the associated calculated equidistant paths. This can result in a "bottleneck" being falsely detected at outside corners. The reason for this is that the calculated tool path does not run equidistant to the programmed original contour when `DISC>0`.

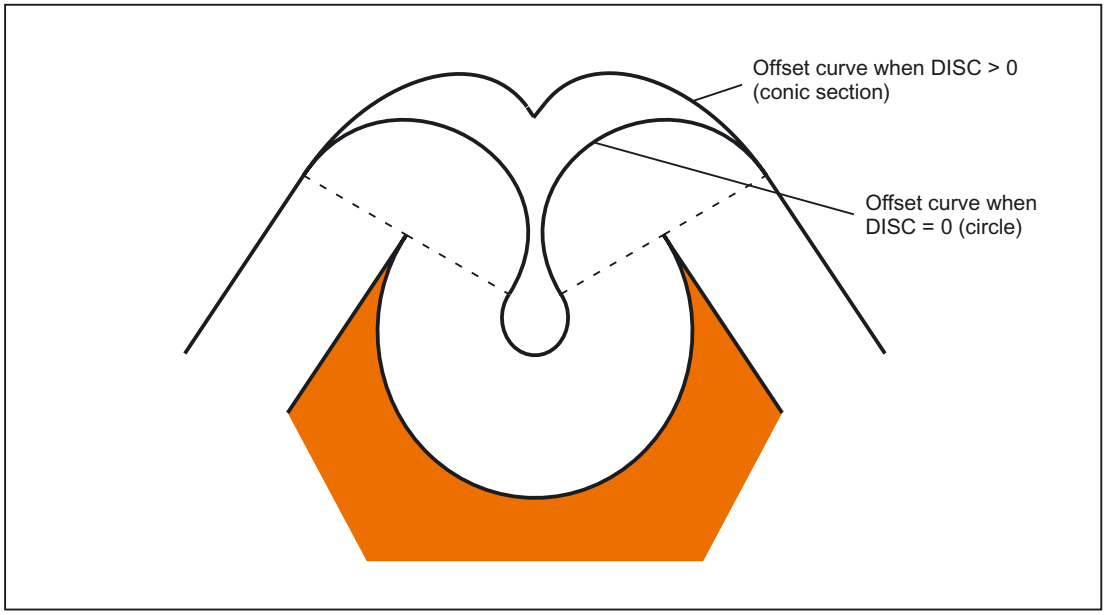


Figure 17-31 Bottleneck detection and outside corners

17.5.9 Blocks with variable compensation value

Supplementary conditions

A variable compensation value is permissible for all types of interpolation (including circular and spine interpolation).

It is also permitted to change the sign (and, therefore, the compensation side).

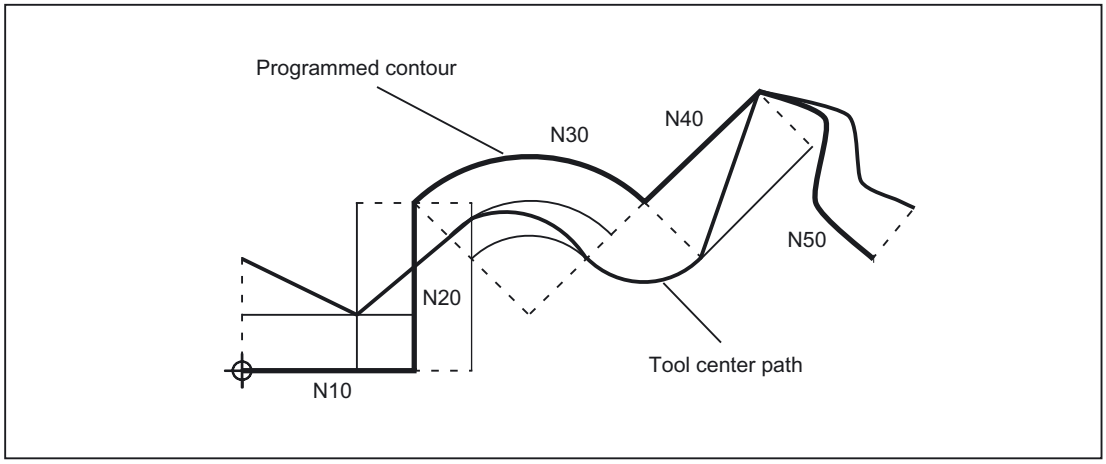


Figure 17-32 Tool radius compensation with variable compensation value

Calculation of intersection

When the intersections in blocks with variable compensation value are calculated, the intersection of the offset curves (tool paths) is always calculated based on the assumption that the compensation value is constant.

If the block with the variable compensation value is the first of the two blocks to be examined in the direction of travel, then the compensation value at the block end is used for the calculation; the compensation value at the block start is used otherwise.

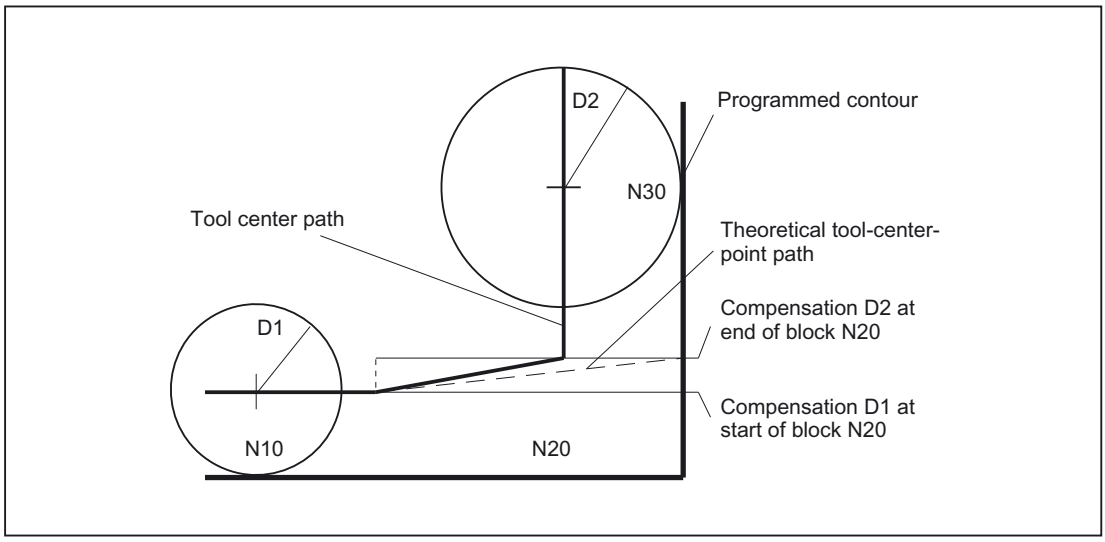


Figure 17-33 Intersection calculation with variable compensation value

Restrictions

If during machining on the inside of the circle the compensation radius becomes geater than the programmed circle radius, then the machining is rejected with the following alarm:

Alarm 10758 "Curvature radius with variable compensation value too small"

Maintain stability of closed contour

If a radius of two circles is increased slightly, a third block may be necessary in order to maintain the stability of the closed contour. This is the case if two adjacent blocks, which represent two possible intersection points for a closed contour, are skipped due to the compensation.

A stable closed contour can be achieved by choosing the first intersection point instead of the second.

SD42496 $SC_CUTCOM_CLSD_CONT ≠ 0 (response of TRC for closed contour).

In that case, the second intersection point is always reached, even if the block is skipped. A third block is then not required.

17.5.10 Keep tool radius compensation constant

Description

The "Keep tool radius compensation constant" function is used to suppress tool radius compensation for a number of blocks, whereby a difference between the programmed and the actual tool center path traveled set up by tool radius compensation in the previous blocks is retained as the compensation.

It can be an advantage to use this method when several traversing blocks are required during line milling in the reversal points, but the contours produced by the tool radius compensation (follow strategies) are not wanted.

Activation

The "Keep tool radius compensation constant" function is activated with the G code CUTCONON (CUTter compensation CONstant ON) and deactivated with the G code CUTCONOF (CUTter compensation CONstant OFF).

CUTCONON and CUTCONOF form a modal G-code group.

The initial setting is CUTCONOF.

The function can be used independently of the type of tool radius compensation ($2^1/_2$D, 3D face milling, 3D circumferential milling).

Normal case

Tool radius compensation is normally active before the compensation suppression and is still active when the compensation suppression is deactivated again.

In the last traversing block before CUTCONON, the offset point in the block end point is approached. All following blocks, in which compensation suppression is active, are traversed without compensation. However, they are offset by the vector from the end point of the last offset block to its offset point. These blocks can have any type of interpolation (linear, circular, polynomial).

The deactivation block of the compensation suppression, i.e. the block that contains CUTCONOF, is compensated normally. It starts in the offset point of the start point. One linear block is inserted between the end point of the previous block, i.e. the last programmed traversing block with active CUTCONON, and this point.

Circular blocks, for which the circle plane is perpendicular to the compensation plane (vertical circles), are treated as though they had CUTCONON programmed. This implicit activation of compensation suppression is automatically canceled in the first traversing block that contains a traversing motion in the compensation plane and is not such a circle. Vertical circle in this sense can only occur during circumferential milling.

Example:

```
N10                                  ; Definition of tool d1
N20 $TC_DP1[1,1] = 110               ; Type
N30 $TC_DP6[1,1]=                    ; Radius
N40
N50 X0 Y0 Z0 G1 G17 T1 D1 F10000
N60
N70 X20 G42 NORM
N80 X30
N90 Y20
N100 X10 CUTCONON                    ; Activate compensation suppression
N110 Y30 KONT                        ; On deactivation of contour suppression,
                                       insert bypass circle, if necessary
N120 X-10 CUTCONOF
N130 Y20 NORM                        ; No bypass circle on deactivation of TRC
N140 X0 Y0 G40
N150 M30
```

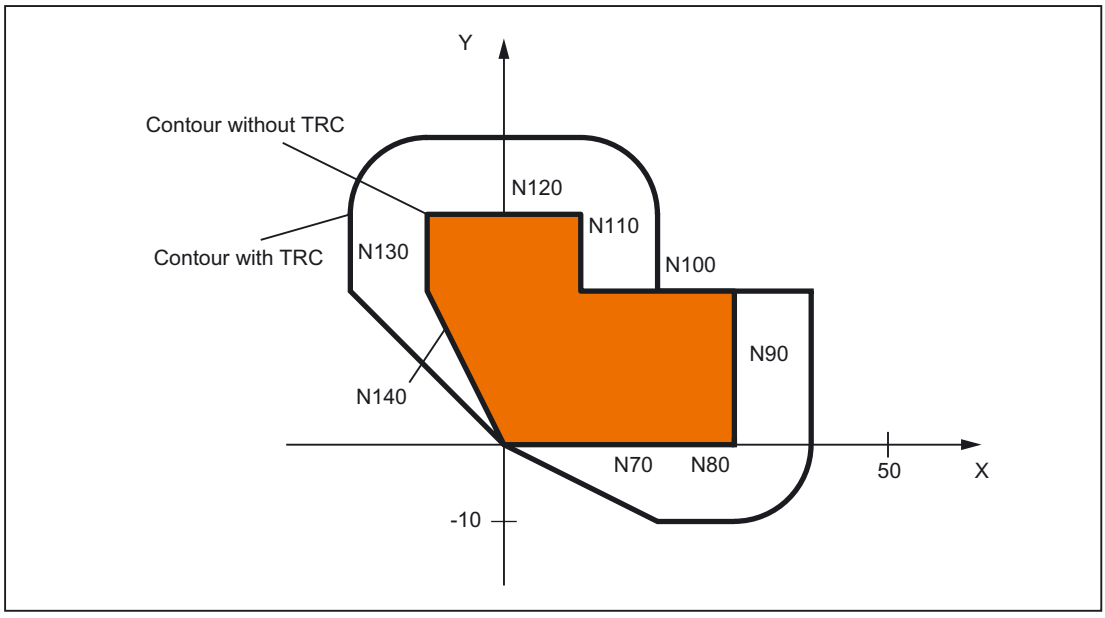


Figure 17-34 Sample program for contour suppression

Special cases

- If tool radius compensation is not active (G40), CUTCONON has no effect. No alarm is produced. The G code remains active, however.

  This is significant if tool radius compensation is to be activated in a later block with G41 or G42.

- It is permissible to change the G code in the 7th G-code group (tool radius compensation; G40 / G41 / G42) with CUTCONON active. A change to G40 is active immediately.

  The offset used for traversing the previous blocks is traveled.

- If CUTCONON or CUTCONOF is programmed in a block without traversing in the active compensation plane, activation is delayed until the next block that has such a traversing motion.

- If CUTCONON is programmed with active tool radius compensation and not canceled before the end of the program, the traversing blocks are traversed with the last valid compensation.

  The same applies for reprogramming of G41 or G42 in the last traversing block of a program.

- If tool radius compensation is activated with G41 or G42 and CUTCONON is also already active, activation of compensation is delayed until the next traversing block with CUTCONOF.

- When reapproaching the contour with CUTCONOF, the 17th G-code group (approach and retraction behavior with tool compensation; NORM / KONT) is evaluated, i.e. a bypass circle is inserted if necessary for KONT. A bypass circle is inserted under the same conditions as for activation of tool radius compensation with G41 or G42.

- The number of blocks with suppressed tool radius compensation is restricted:

  MD20252 $MC_CUTCOM_MAXNUM_SUPPR_BLOCKS (Maximum number of blocks with compensation suppression).

  If it is exceeded, machining is aborted and an error message issued.

  The restriction is necessary because the internal block processing in the last block before CUTCONON must be resumed when repositioning.

- The response after reprogramming G41 or G42 when tool radius compensation is already active is similar to compensation suppression.

  The following deviations apply:

  – Only linear blocks are permissible

  – A single traversing block that contains G41 or G42 is modified so that it ends at the offset point of the start point in the following block. Thus it is not necessary to insert a dummy block. The same applies to the last block in a sequence of traversing blocks where each contains G41 or G42.

  – The contour is always reapproached with NORM, independent of the G code of the 17th group (approach and retraction behavior with tool compensation; NORM / KONT).

- If G41 / G42 is programmed several times in consecutive traversing blocks, all blocks are machined as for CUTCONON, except for the last one.
- The type of contour suppression is evaluated only in the first traversing block of a sequence of consecutive traversing blocks.

  If both CUTCONON and G41 or G42 are programmed in the first block, the response to deactivating contour suppression is determined by CUTCONON.

  Changing from G41 to G42 or vice versa makes sense in this case as a means of changing the compensation side (left or right of the contour) when restarting.

  A change of compensation side (G41/G42) can also be programmed in a later block, even if contour suppression is active.
- Collision detection and bottleneck detection is deactivated for all blocks with active contour suppression.

17.5.11 Alarm behavior

Alarm during preprocessing

If a tool radius compensation alarm is output during preprocessing, main-run machining stops at the next block end reached, i.e. usually at the end of the block currently being interpolated (if Look Ahead is active, once the axes have come to a stop).

Alarms for preprocessing stop and active tool radius compensation

Tool radius compensation generally requires at least one of the following traversing blocks (even more for bottlenecks) to determine the end point of a block. Since the preprocessing stop of such a block is not available, traversing continues to the offset point in the last block. Correspondingly, the offset point in the start point is approached in the first block after a preprocessing stop.

The contour obtained may deviate considerably from the one that would result without preprocessing stop. Contour violations in particular are possible. Therefore the following setting data was introduced:

SD42480 $MC_STOP_CUTCOM_STOPRE (alarm response for TRC and preprocessing stop).

The response of the tool radius compensation remains unchanged compared to the previous status, and/or an alarm is output for preprocessing stop during active tool radius compensation and the program is halted, depending on the value.

The user can acknowledge this alarm and continue the NC program with NC start or abort it with RESET.

17.5.12 Intersection procedure for polynomials

Function

If two curves with active tool radius compensation form an outside corner, depending on the G code of the 18th group (corner behavior with tool compensation; G450 / G451) and regardless of the type of curves involved (straight lines, circles, polynomials):

- Either a conic is inserted to bypass the corner

  Or

- The curves involved are extrapolated to form an intersection.

If no intersection is found with G451 activated, or if the angle formed by the two curves is too steep, switchover to insert mode is automatic.

The intersection procedure for polynomials is released with the machine data:

MD20256 $MC_CUTCOM_INTERS_POLY_ENABLE (Intersection process possible for polynomials)

Note

If this machine data is set to inactive, a block (can be very short) is always inserted (even if transitions are almost tangential). These short blocks always produce unwanted drops in speed during G64 operation.

17.5.13 G461/G462 Approach/retract strategy expansion

Function

In certain special geometrical situations, extended approach and retraction strategies, compared with the previous implementation, are required in order to activate or deactivate tool radius compensation (see figure below).

Note

The following example describes only the situation for deactivation of tool radius compensation. The response for approach is virtually identical.

Example

```
G42 D1 T1                          ; Tool radius 20 mm
...
G1 X110 Y0
N10 X0
N20 Y10
N30 G40 X50 Y50
```

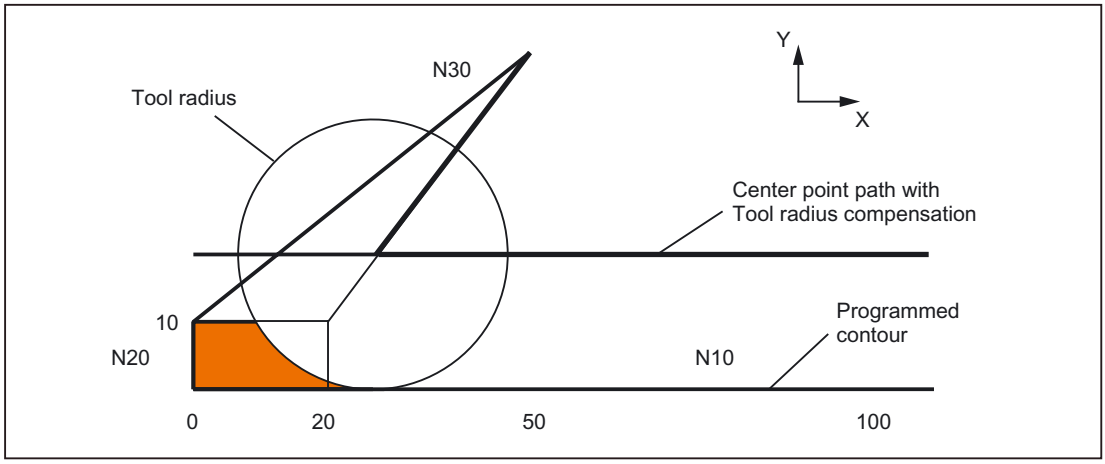


Figure 17-35 Retraction behavior with G460

The last block with active tool radius compensation (N20) is so short that an intersection no longer exists between the offset curve and the preceding block (or a previous block) for the current tool radius. An intersection between the offset curves of the following and preceding blocks is therefore sought, i.e., between N10 and N30 in this example. The curve used for the retraction block is not a real offset curve, but a straight line from the offset point at the end of block N20 to the programmed end point of N30. The intersection is approached if one is found. The colored area in the figure is not machined, although the tool used would be capable of this.

G460

With G460, the approach/retraction strategy is the same as before.

G461

If no intersection is possible between the last TRC block and a preceding block, the offset curve of this block is extended with a circle whose center point lies at the end point of the uncorrected block and whose radius is equal to the tool radius.

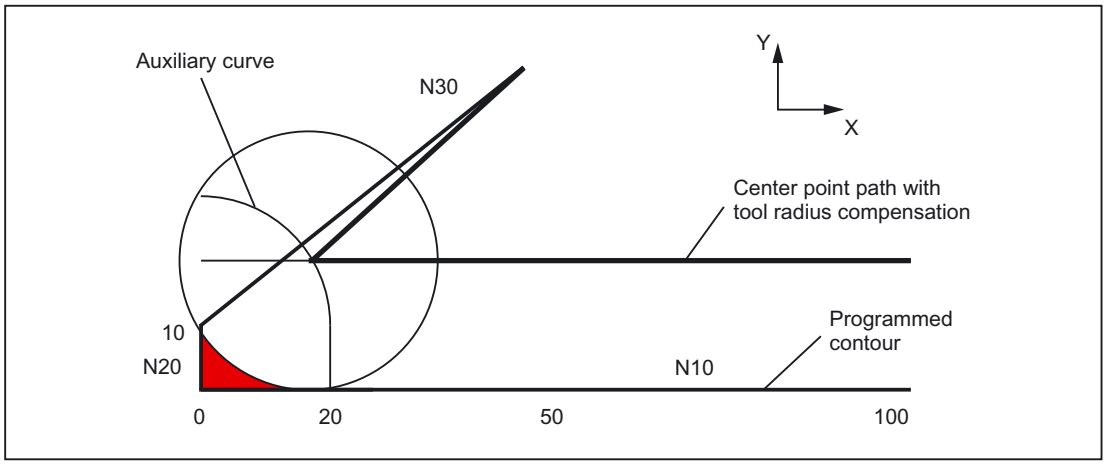


Figure 17-36 Retraction behavior with G461

The control attempts to cut this circle with one of the preceding blocks. If CDOF is active, the search is terminated when an intersection is found, i.e., the system does not check for more intersections with even earlier blocks.

If CDON is active, the search for more intersections continues after the first intersection is found.

An intersection point, which is found in this way, is the new end point of a preceding block and the start point of the deactivation block. The inserted circle is used exclusively to calculate the intersection and does not produce a traversing movement.

Note

If no intersection is found, the following alarm is output:

Alarm "10751 Collision danger"

G462

If no intersection is possible between the last TRC block and a preceding block, a straight line is inserted, on retraction with G462 (initial setting), at the end point of the last block with tool radius compensation (the block is extended by its end tangent).

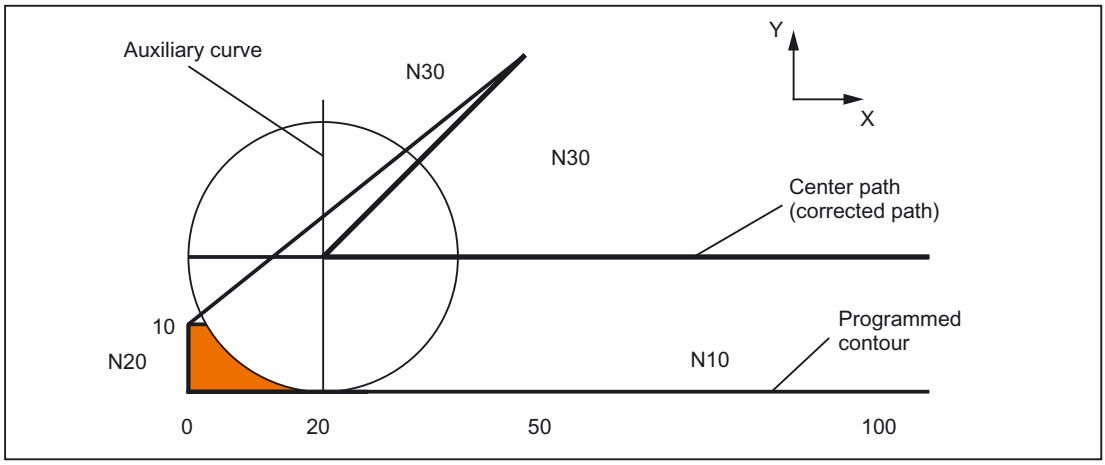


Figure 17-37 Retraction behavior with G462

The search for the intersection is then identical to the procedure for G461.

With G462, the corner generated by N10 and N20 in the sample program is not machined to the full extent actually possible with the tool used. However, this behavior may be necessary if the part contour (as distinct from the programmed contour), to the left of N20 in the example, is not permitted to be violated even with y values greater than 10 mm.

If KONT is active (travel round contour at start or end point), behavior will differ according to whether the end point is in front of or behind the contour.

End point in front of contour

If the end point is located in front of the contour, the retraction behavior is the same as for NORM. This feature does not change, even if the last contour block with G451 is extended with a straight line or a circle. Additional circumnavigation strategies to avoid a contour violation in the vicinity of the contour end point are therefore not required.

End point behind contour

If the end point is behind the contour, a circle or straight line is always inserted depending on G450/G451. In this case, G460-G462 has no effect.

If, in this situation, the last traversing block has no intersection with a preceding block, an intersection with the inserted contour element or with the linear section from the end point of the circumnavigation circle to the programmed end point can result.

If the inserted contour element is a circle (G450), and it intersects with the preceding block, this is the same as the intersection, which would be produced with NORM and G461. In general, however, a remaining section of the circle still has to be traversed. An intersection calculation is no longer required for the linear section of the retraction block.

In the second case (if no intersection is found between the inserted contour element and the preceding blocks), the intersection between the retraction straight line and a preceding block is approached.

Therefore, when G461 or G462 is active, behavior deviating from G460 can only arise if NORM is active or if behavior with KONT is identical to NORM due to the geometrical conditions.

Note

The approach behavior is symmetrical to the retraction behavior.

The approach/retraction behavior is determined by the state of the G command in the approach/retraction block. The approach behavior can therefore be set independently of the retraction behavior.

Example:

Program for using G461 during approach:

```
N10 $TC_DP1[1,1]=120                    ; Tool type mill
N20 $TC_DP6[1,1]=10                     ; Radius
N30 X0 Y0 F10000 T1 D1
N40 Y20
N50 G42 X50 Y5 G461
N60 Y0 F600
N70 X30
N80 X20 Y-5
N90 X0 Y0 G40
N100 M30
```

17.6 Toolholder with orientation capability

17.6.1 General

Introduction

The orientation of the tool can vary (e.g. **due to retooling**) for one single class of machine tools. When the machine is operating, the orientation that has been set is **permanent**, however, and cannot be changed during traversing. For this reason, kinematic orientation transformation (3-, 4- or 5axis transformations, `TRAORI`) is neither necessary nor does it make sense for such machines. However, it is necessary to take account of the changes in the tool length components caused by changing the orientation, without having to trouble the user with mathematics involved. The control performs these calculations.

Availability

For SINUMERIK 828D, the "toolholder with orientation capability" function is only available for the milling versions.

Required data

The following requirements must be met if the control is to take tool compensations into account for toolholders with orientation capability:

- Tool data (geometry, wear, etc.)
- Toolholder data (data for the geometry of the toolholder with orientation capability)

Toolholder selection

A toolholder defined in the control must be specified for the "Toolholder with orientation capability" function. The NC program command below is used for this purpose:

```
TCARR = m
```

m: Number of the toolholder

The toolholder has an associated toolholder data block, which describes its geometry.

Activating the toolholder and its block has an immediate effect, i.e. from the next traversing block onwards.

Assignment tool/toolholder

The tool that was active previously is assigned to the new toolholder.

From the point of view of the control, toolholder number m and tool numbers T can be combined freely. In the real application, however, certain combinations can be ruled out for machining or mechanical reasons. The control does not check whether the combinations make sense.

Description of the kinematics of the toolholder

The kinematics of the toolholder with orientation capability is described with a total of 33 parameter sets.

The data of the data block can be edited by the user.

Toolholder with orientation capability

Example: Cardan toolholder with two axes for the tool orientation

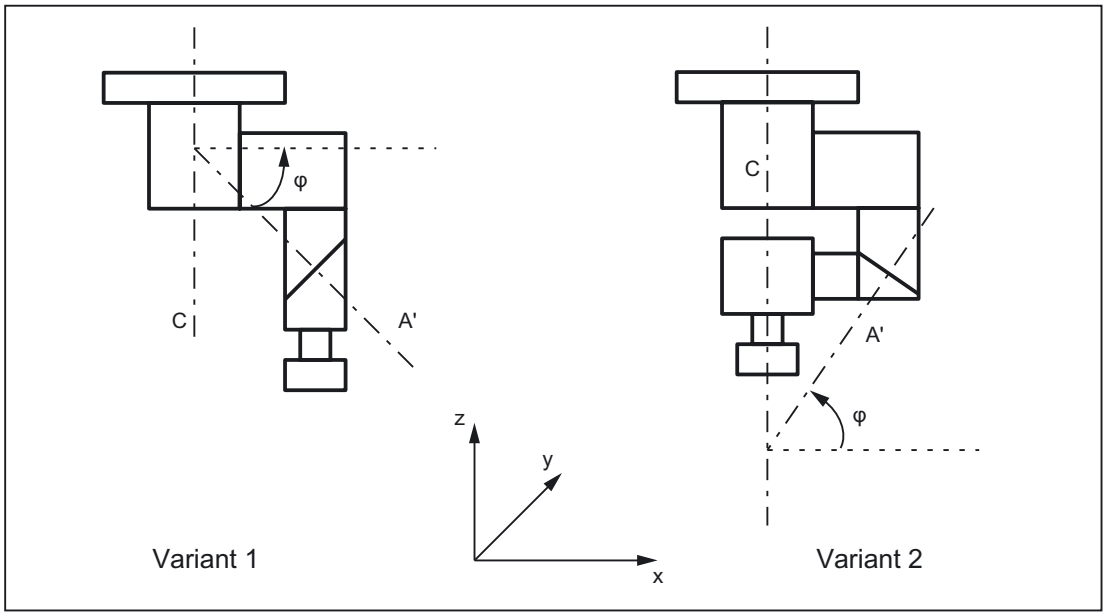


Figure 17-38 Cardan toolholder with two axes

Processing toolholder data blocks

Two options are available:

- Explicit entry in the toolholder data block from the part program
- Automatic acceptance of certain values (angles) from a frame

  A requirement for this is that TCOFR (Tool Carrier Orientation FRame) is also specified when the toolholder is selected.

  The tool orientation used to calculate the tool length is determined again from the frame active at this time when a toolholder is changed.

Orientation in Z direction

The G function TOFRAME defines a frame such that the Z direction in this frame is the same as the current tool orientation.

If no tool carrier or a tool carrier without change in orientation is active, then the Z direction is in the new frame:

- The same as the old Z direction with G17.
- The same as the old Y direction with G18.
- The same as the old X direction with G19.

TCOABS for active frame

The absolute toolholder orientation is set using:

TCOABS (Tool Carrier Orientation ABSolute)

The orientation taken into account for the tool length compensation is **independent** of the orientation of the active frame.

Only one of the instructions TCOABS or TCOFR can be valid.

Frame change

The user can change the frame following selection of the tool. This does not have any effect on the tool length compensation components.

Angles in the toolholder data:

The programmed angles of rotation stored in the toolholder data are not affected by the angle of rotation defined by the frames. When changing from TCOFR to TCOABS, the original (programmed) angles of rotation in the toolholder data are reactivated.

Tool compensation types

TRC takes account of the current tool orientation when CUT2D or CUT3DFS is active.

All other tool compensation types

These are all the compensation types of G-code group 22, with the exception of CUT3DC and CUT3DF. The response remains the same with respect to the plane used for compensation. This is determined independent of the tool orientation from the active frame.

For CUT2DF and CUT3DFF, the compensation plane used for TRC is determined from the frame **independently** of the current tool orientation. The active plane (G17/G18/G19) is considered.

CUT3DC and CUT3DF

3D tool compensation for circumferential milling

3D tool compensations for face milling with active 5-axis transformation are not affected by the "Toolholder with orientation capability" function.

The orientation information is determined by the active kinematic 5-axis transformation.

Limited toolholder orientation

An alarm is output if an orientation that cannot be reached with the defined toolholder kinematic is specified by the frame.

The following kinematics cannot achieve any orientation:

- If the two rotary axes which are necessary to define the kinematics are not perpendicular to each other and if the tool axis which defines the tool direction is not perpendicular to the second rotary axis

  or

- Fewer than two axes have been defined

Non-rotary toolholders

The tool orientation used internally is dependent only on the basic orientation of the tool and the active plane (G17 - G19).

Ambiguities

With two axes, a particular tool orientation defined by the frame can generally be set with **two** different rotary angle pairs. Of these two, the control selects the setting with which the rotary angle is as close as possible to the programmed rotary angle.

Storing angles in the toolholder data

In virtually any case where ambiguities may arise, it is necessary to store the approximate angle expected from the frame in the toolholder data.

Parameter sets

A complete set of parameters for a toolholder with orientation capability consists of 33 values.

The following system variables are available:

- $TC_CARR1 to $TC_CARR33
- In addition, $TC_CARR34 to $TC_CARR65 are freely available for the user and for fine offsets.

The significance of the individual parameters is distinguished as follows:

Machine kinematics:

$TC_CARR1 to $TC_CARR20 and $TC_CARR23

$TC_CARR18 to $TC_CARR20 define a further vector l_4, which is needed to describe the machine with extended kinematics (both tool and workpiece can be rotated).

$TC_CARR21 and $TC_CARR22 contain the channel-axis identifiers of the rotary axes, the positions of which can be used to determine the orientation of the toolholder with orientation capability, if necessary.

Kinematic type:

$TC_CARR23 using letter T, P or M

The following three options are available for the kinematic type, for which both upper and lower case text are permissible:

T:	Only the (**T**ool) can be rotated (basic value).
P:	Only the workpiece (**P**art) can be rotated.
M:	Both tool and workpiece can be rotated (**M**ixed mode).

Any character other than the three mentioned here will result in the alarm if it is tried to activate the toolholder with orientation capability:

Alarm "14153 Channel %1 block %2 unknown tool carrier type: %3"

Rotary axis parameters:

$TC_CARR24 to $TC_CARR33

The system variables in $TC_CARR24 to $TC_CARR33 can be used to define offsets, angle compensations, Hirth tooth system and axis limits.

Note

The system variables are available with and without active tool management.

Components and presetting of the chain/data block

The values $TC_CARR1 to $TC_CARR20 and $TC_CARR24 to $TC_CARR33 in the toolholder data block are of NC language format type `REAL`..

The values $TC_CARR21 and $TC_CARR22 for the axis identifier of the first rotary axis (v_1) and the second rotary axis (v_2) are of NC language format type AXIS. They are all preset to zero.

The value $TC_CARR23 is initialized with the uppercase letter "T" (only tool can be rotated).

$TC_CARRn[m]

$TC_CARR[0]= 0 has a special significance

System variables for toolholders with orientation capability

$TC_CARRn[m]

n: Parameters 1...33

m: Number of the tool carrier 1 that can be oriented...Value of the machine data:
MD18088 $MN_MM_NUM_TOOL_CARRIER (maximum number of definable tool carriers)

Description	NCK variable	Language format	Default setting
x component of offset vector l_1	$TC_CARR1	REAL	0
y component of offset vector l_1	$TC_CARR2	REAL	0
z component of offset vector l_1	$TC_CARR3	REAL	0
x component of offset vector l_2	$TC_CARR4	REAL	0
y component of offset vector l_2	$TC_CARR5	REAL	0
z component of offset vector l_2	$TC_CARR6	REAL	0
x component of rotary axis v_1	$TC_CARR7	REAL	0
y component of rotary axis v_1	$TC_CARR8	REAL	0
z component of rotary axis v_1	$TC_CARR9	REAL	0
x component of rotary axis v_2	$TC_CARR10	REAL	0
y component of rotary axis v_2	$TC_CARR11	REAL	0
z component of rotary axis v_2	$TC_CARR12	REAL	0
Angle of rotation α1 (in degrees)	$TC_CARR13	REAL	0
Angle of rotation α_2 (in degrees)	$TC_CARR14	REAL	0
x component of offset vector l_3	$TC_CARR15	REAL	0
y component of offset vector l_3	$TC_CARR16	REAL	0
z component of offset vector l_3	$TC_CARR17	REAL	0
x component of offset vector l_4	$TC_CARR18	REAL	0
y component of offset vector l_4	$TC_CARR19	REAL	0
z component of offset vector l_4	$TC_CARR20	REAL	0
Axis identifier of the rotary axis v_1	$TC_CARR21	AXIS	0
Axis identifier of the rotary axis v_2	$TC_CARR22	AXIS	0
Kinematic type	$TC_CARR23	CHAR	T
Offset of rotary axis v_1	$TC_CARR24	REAL	0
Offset of rotary axis v_2	$TC_CARR25	REAL	0
Angle offset of rotary axis v_1 (Hirth tooth)	$TC_CARR26	REAL	0
Angle offset of rotary axis v_2 (Hirth tooth)	$TC_CARR27	REAL	0
Angle increment of rotary axis v_1 (Hirth tooth)	$TC_CARR28	REAL	0
Angle increment of rotary axis v_2 (Hirth tooth)	$TC_CARR29	REAL	0
Minimum position of rotary axis v_1 (SW limit)	$TC_CARR30	REAL	0
Minimum position of rotary axis v_2 (SW limit)	$TC_CARR31	REAL	0
Maximum position of rotary axis v_1 (SW limit)	$TC_CARR32	REAL	0
Maximum position of rotary axis v_2 (SW limit)	$TC_CARR33	REAL	0

System variables for the user and for fine offsets

- $TC_CARR34 to $TC_CARR40

  Contain parameters, which are freely available to the user.

- $TC_CARR41 to $TC_CARR65

  Contain fine offset parameters that can be added to the values in the basic parameters. The fine offset value assigned to a basic parameter is obtained when the value 40 is added to the parameter number.

- $TC_CARR47 to $TC_CARR54 and $TC_CARR61 to $TC_CARR63

  Not defined and produce an alarm if read or write access is attempted.

Description	NCK variable	Language format	Default setting
Toolholder name *	$TC_CARR34	String[32]	""
Axis name 1 **	$TC_CARR35	String[32]	""
Axis name 2 **	$TC_CARR36	String[32]	""
Identifier **	$TC_CARR37	INT	0
Position component X **	$TC_CARR38	REAL	0
Position component Y **	$TC_CARR39	REAL	0
Position component Z **	$TC_CARR40	REAL	0
x comp. fine offset of offset vector l_1	$TC_CARR41	REAL	0
y comp. fine offset of offset vector l_1	$TC_CARR42	REAL	0
z comp. fine offset of offset vector l_1	$TC_CARR43	REAL	0
x comp. fine offset of offset vector l_2	$TC_CARR44	REAL	0
y comp. fine offset of offset vector l_2	$TC_CARR45	REAL	0
z comp. fine offset of offset vector l_2	$TC_CARR46	REAL	0
x comp. fine offset of offset vector l_3	$TC_CARR55	REAL	0
y comp. fine offset of offset vector l_3	$TC_CARR56	REAL	0
z comp. fine offset of offset vector l_3	$TC_CARR57	REAL	0
x comp. fine offset of offset vector l_4	$TC_CARR58	REAL	0
y comp. fine offset of offset vector l_4	$TC_CARR59	REAL	0
z comp. fine offset of offset vector l_4	$TC_CARR60	REAL	0
Offset of fine offset of rotary axis v_1	$TC_CARR64	REAL	0
Offset of fine offset of rotary axis v_2	$TC_CARR65	REAL	0
Remarks:			
*	A toolholder with orientation capability cannot subsequently be assigned a number with system variable $TC_CARR34, only a name.		
**	System variables $TC_CARR35 to $TC_CARR40 refer to the intended use of the toolholder with orientation capability within the measuring cycles and can also be used for other purposes.		

17.6.2 Kinematic interaction and machine design

Representation of the kinematic chain

The concept of the kinematic chain is used to describe the kinematic interaction between a reference point and the tool tip.

The chain specifies all the data required for the toolholder data block in a schematic. To describe the concrete case with a particular kinematic, the relevant components of the chain must be assigned real vectors, lengths and angles. The chain represents the maximum constellation. In simpler applications, individual components can be zero (e.g. kinematics with one or no rotary axis).

The machine does not have to have axes that rotate the tool and/or workpiece table. The function can be used even if the orientations are set manually by handwheels or reconfiguration.

The machine design is described by the following parameters:

- Two rotary axes (v_1 and v_2), each with one angle of rotation (α_1 or α_2), which counts positively for clockwise rotation facing the direction of the rotation vector.
- Up to four offset vectors (l_1 to l_4) for relevant machine dimensions (axis distances, distances to machine or tool reference points).

Zero vectors

Vectors v_1 and v_2 can be zero. The associated angle of rotation (explicitly programmed or calculated from the active frame) must then also be zero, since the direction of the rotating axis is not defined. If this condition is not satisfied, an alarm is produced when the toolholder is activated.

Less than two rotating axes

The option not to define a rotating axis makes sense when the toolholder to be described can only rotate the tool in one plane. A sensible minimum data block may, therefore, contain only one single entry not equal to 0 in the toolholder data; namely, a value in one of the components of v_1 or v_2 for describing a rotating axis parallel to the axis where the angle of rotation α1 or α2 is determined from one frame.

Further special cases

Vectors v_1 and v_2 can be colinear. However, the degree of freedom for orientation is lost, i.e. this type of kinematic is the same as one where only one rotary axis is defined. All possible orientations lie on one cone sheath. The conical sheath deforms to a straight line if tool orientation t and v_1 or v_2 become colinear. Change of orientation is, therefore, not possible in this special case. The cone sheath deforms to a circular surface (i.e. all orientations are possible in one plane), if tool orientation t and v_1 or v_2 are perpendicular to each other.

It is permissible for the two vectors v_1 and v_2 to be zero. A change in orientation is then no longer possible. In this special case, any lengths l_1 and l_2, which are not equal to zero, act as additional tool length compensations, in which the components in the individual axes are not affected by changing the plane (G17 - G19).

Kinematics data expansions

- Possibility of direct access to existing machine axes in order to define the toolholder setting via the rotary axis positions.
- Extension of the kinematics with rotary workpiece and on kinematics with rotary tool and rotary workpiece.
- Possibility to permit only discrete values in a grid for the rotary axis positions (Hirth tooth system).

The extensions are compatible with earlier software versions and encompass the kinematic data blocks from $TC_CARR18 to $TC_CARR23.

Machine with rotary tool

On machines with rotary tool there is no change in the definition of the kinematics compared to older software versions. The newly introduced vector l_4, in particular, has no significance. Should the contents of l_4 not be zero, this is ignored.

The term "Toolholder with orientation capability" is actually no longer really appropriate for the new kinematic types, with which the table can also be rotated, either alone or additionally to the tool. However, it has been kept for reasons of compatibility.

The kinematic chains used to describe the machine with rotary tool (general case) are shown in the figure below:

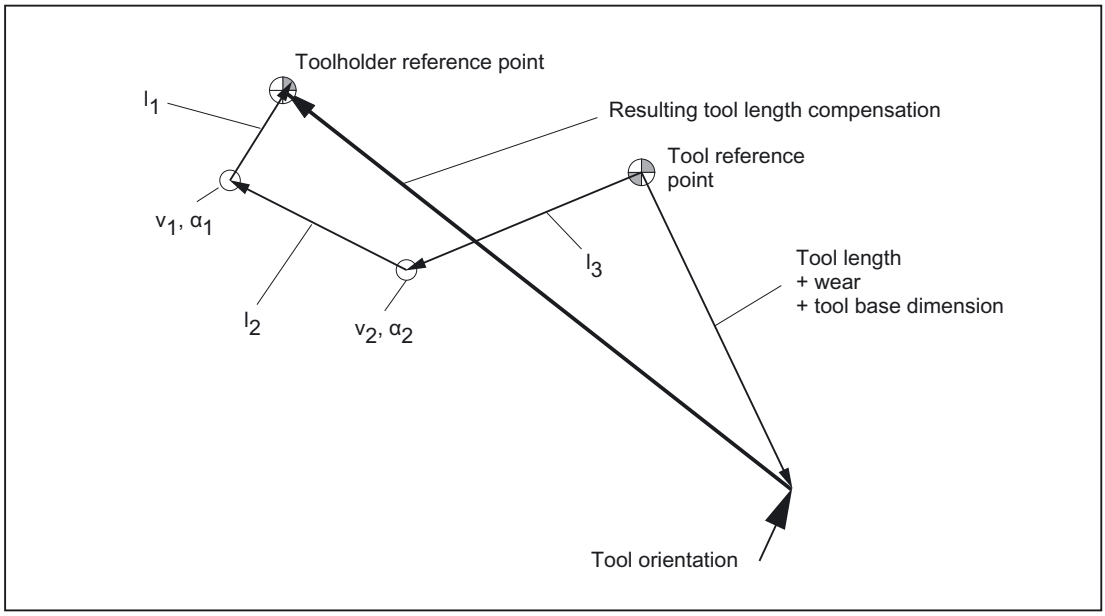


Figure 17-39 Kinematic chain to describe a tool with orientation

Vectors, which describe offsets in the rotary head, are positive in the direction from the tool tip to the reference point of the toolholder.

The following kinematic type is defined for machines with a rotary tool:

$TC_CARR23 using letter T

Machine with rotary workpiece

On machines with rotary workpiece, the vector l_1 has no significance. If it contains a value other than zero, this is ignored.

The kinematic chain for machines with rotary workpiece is shown in the figure below.

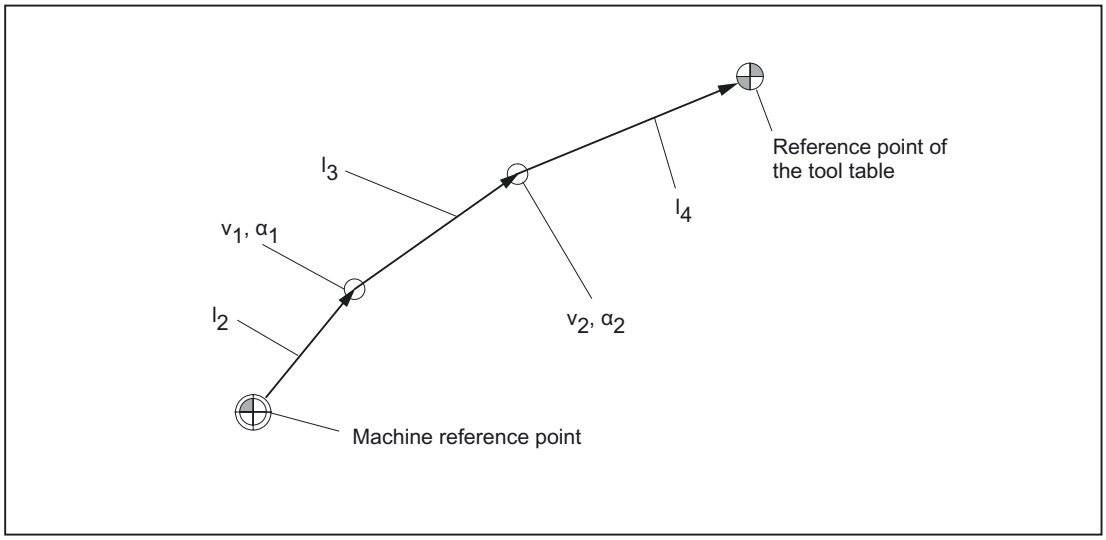


Figure 17-40 Kinematic chain to describe a rotary table

Vectors, which describe offsets in the rotary table, are positive in the direction from the machine reference point to the table.

The following kinematic type is defined for machines with a rotary workpiece:

$TC_CARR23 using letter P

Note

On machines with rotary workpiece it is generally useful if the selected machine reference point and the reference point of the table are identical. Selecting the reference points in this way has the advantage that the position of the workpiece zero in the initial state (i.e. with rotary axes not turned) does not change when the rotary table is activated. The (open) kinematic chain (see figure) is then closed.

In this special case, therefore, the following formula applies: $l_2 = - (l_3 + l_4)$

Machines with extended kinematics

On machines with extended kinematics (both tool and workpiece are rotary), it is only possible to turn each of the components with one axis.

The kinematic of the rotary tool is described with the first rotary axis (v_1) and the two vectors l_1 and l_2, that of the rotary table with the second rotary axis (v_2) and the two vectors l_3 and l_4. The two kinematic chain components for machines with rotary tool and rotary workpiece are shown in the figure below.

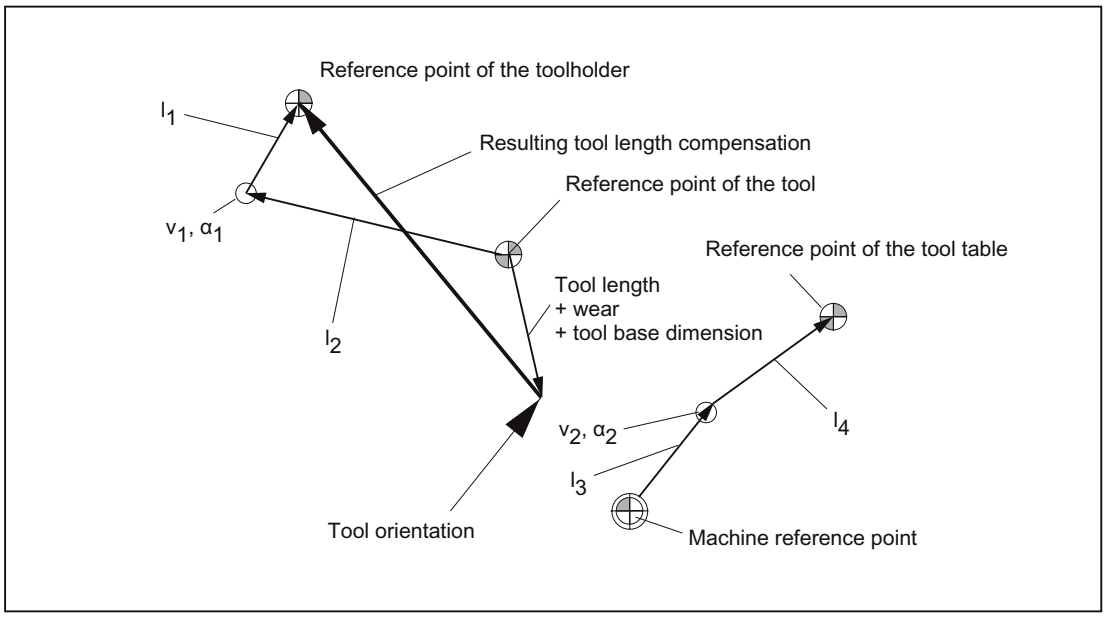


Figure 17-41 Kinematic sequence with extended kinematics

The following kinematic type is defined for machines with a rotary tool and rotary workpiece:

$TC_CARR23 using letter M (extended kinematics)

Note

On machines with extended kinematics it is generally useful, as with machines where only the table can be rotated, for the machine reference point and the reference point of the table to be identical. The (open) chain component to describe the table (see figure) is then closed.

In this special case, the following formula applies: $l_3 = - l_4$

Rotary tool types T and M

For machine kinematics with a rotary tool (types T and M), the toolholder component with orientation capability, which describes the tool or head component (as opposed to the table component), acts, in conjunction with the active tool, as a new overall tool.

Fine offset

The offset vectors l_1 to l_4 and the offsets of the rotary axes v_1 and v_2 can be represented as the sum of a basic value and a fine offset. The fine offset parameters assigned to the basic values are achieved by **adding a value of 40 to the index of the basic value**.

Example:

The parameter $TC_CARR5 is assigned to the fine offset $TC_CARR45.

Note

For the significance of the system variables $TC_CARR41 to $TC_CARR65 available for the fine offset see:

References:
Programming Manual, Job Planning; Tool Offsets:

Activation

The following setting adds the fine offset values to the basic values:

SD42974 $SC_TOCARR_FINE_CORRECTION = 1 (fine offset TCARR on/off)

Supplementary conditions

The amount is limited to the permissible fine offset.

The maximum permissible value is defined:

For:	With machine data:
• The components of vectors l_1 to l_4:	MD20188 $MC_TOCARR_FINE_LIM_LIN
• The offsets of the two rotary axes v_1 and v_2:	MD20190 $MC_TOCARR_FINE_LIM_ROT

An illegal fine offset value is only detected when:

- A toolholder with orientation capability, which contains such a value, is activated

  and

- at the same time the following setting data is set:

  SD42974 $SC_TOCARR_FINE_CORRECTION

Description of a rotation

A data block for describing a rotation comprises one vector v_1 /v_2 to describe the direction of rotation of the rotary axis in its initial state and an angle α1/α2. The angle of rotation is counted positively for clockwise rotation facing the direction of the rotation vector.

The two toolholder angles α1 and α2 are determined using a frame, independent of the active plane currently selected (G17 - G19).

The tool orientation in the initial state (both angles α1 and α2 are zero) is (as in the default case):

- G17: Parallel to Z.
- G18: Parallel to Y.
- G19: Parallel to Z

Assigning data to the toolholder

Example of a machine with rotary toolholder

The following settings are obtained at the mill head shown for a machine with toolholder with orientation capability of kinematic type T:

Component of the offset vector l_1 =	(-200, 0, 0)
Component of the offset vector l_2 =	(0, 0, 0)
Component of the offset vector l_3 =	(-100, 0, 0)
Component of rotary axis v_1 =	(1, 0, 0)
Component of rotary axis v_2 =	(-1, 0, 1)
Tool base dimension of tool reference point	(0, 0, 250)

Note

The tool reference point for the tool base dimension is defined by the reference point at the machine.

For further information about the reference points in the working area, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 721)".

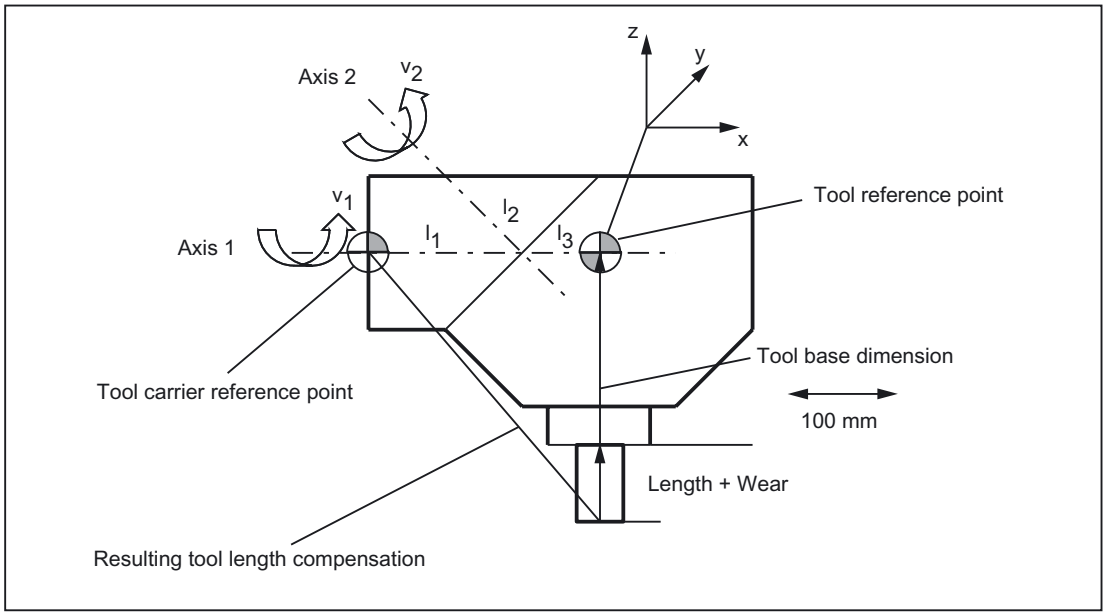


Figure 17-42 Assignment of the toolholder data

Suitable assumptions were made for the following values in the data block:

- The two rotary axes intersect at one point.

  All components of l_2 are therefore zero.

- The first rotary axis lies in the x/z plane, the second rotary axis is parallel to the x axis.

  These conditions define the directions of v_1 and v_2 (the lengths are irrelevant, provided that they are not equal to zero).

- The reference point of the toolholder lies 200 mm in the negative x direction viewed from the intersection of the two rotary axes.

  This condition defines l_1 .

Specify associated data block values

The following associated data block values are specified for the toolholder shown on a machine with rotary toolholder:

Description	NCK variable	Value
x component of offset vector l_1	$TC_CARR1	- 200
y component of offset vector l_1	$TC_CARR2	0
z component of offset vector l_1	$TC_CARR3	0
x component of offset vector l_2	$TC_CARR4	0
y component of offset vector l_2	$TC_CARR5	0
z component of offset vector l_2	$TC_CARR6	0
x component of rotary axis v_1	$TC_CARR7	1
y component of rotary axis v_1	$TC_CARR8	0
z component of rotary axis v_1	$TC_CARR9	0
x component of rotary axis v_2	$TC_CARR10	-1
y component of rotary axis v_2	$TC_CARR11	0
z component of rotary axis v_2	$TC_CARR12	1
Angle of rotation α1 (in degrees)	$TC_CARR13	0
Angle of rotation α2 (in degrees)	$TC_CARR14	0
x component of offset vector l_3	$TC_CARR15	-100
y component of offset vector l_3	$TC_CARR16	0
z component of offset vector l_3	$TC_CARR17	0

Explanations

The toolholder kinematic chosen in the example is such that the two rotary axes form an angle of 45 degrees, which means that the orientation cannot take just any value. In concrete terms, this example does not permit the display of orientations with negative X components.

x component of the tool base dimension:	0
y component of the tool base dimension:	0
z component of the tool base dimension:	250

Note

The required data cannot be determined unequivocally from the geometry of the toolholder, i.e. the user is free to a certain extent to decide the data to be stored. Thus, for the example, it is possible to specify only one z component for the tool base dimension up to the second axis. In this case, l_2 would no longer be zero, but would contain the components of the distance between this point on the second axis and a further point on the first axis. The point on the first axis can also be selected freely. Depending on the point selected, l_1 must be selected such that the reference point (which can also be selected freely) is reached.

In general: vector components that are not changed by rotation of an axis can be distributed over any vectors "before" and "after" rotation.

17.6.3 Inclined surface machining with 3 + 2 axes

Description of function

Inclined machining with 3 + 2 axes describes an extension of the concept of toolholders with orientation capability and applies this concept to machines with a rotary table, on which the orientation of tool and table can be changed simultaneously.

The "Inclined machining with 3 + 2 axes" function is used to machine surfaces with any rotation with reference to the main planes X/Y (G17), Z/X (G18) and Y/Z (G19).

It is possible to produce any orientation of the tool relative to the workpiece by rotating either the tool, the workpiece or both the tool and the workpiece.

The software automatically calculates the necessary compensating movements resulting from the tool lengths, lever arms and the angle of the rotary axis. It is always assumed that the required orientation is set first and not modified during a machining process such as pocket milling on an inclined plane.

Furthermore, the following 3 functions are described, which are required for oblique machining:

- **Position programming** in the direction of the tool orientation independent of an active frame
- Definition of a **frame rotation** by specifying the solid angle
- Definition of the **component of rotation in tool direction** in the programmed frame while maintaining the remaining frame components

Demarcation to 5-axis transformation

If the required functionality specifies that the TCP (Tool Center Point) does not vary in the event of reorientation with reference to the workpiece, even during interpolation, the 5axis software is required.

For more explanations on 5-axis transformations, see:

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

Specification of the toolholder with orientation capability

The toolholder with orientation capability is represented by a general 5axis kinematic sequence described by a data block in the tool compensation memory with a total of 33 REAL values. For toolholders that have two rotary axes for setting the orientation (e.g. a millhead), 31 of these values are constant.

In the current SW version, a data block in the tool compensation memory is described with a total of 47 REAL values. For toolholders that have two rotary axes for setting the orientation, 45 of these values are constant.

The remaining two values are variable and are used to specify the orientation. The constant values describe offsets and directions and setting options for the rotary axes; the variable values describe the angles of the rotary axes.

17.6.4 Machine with rotary work table

System variables

To date, the angles stored in $TC_CARR13 and $TC_CARR14 were used for the calculation of the active tool length with `TCOABS`. This still applies if $TC_CARR21 and $TC_CARR22 do not refer to rotary axes. If $TC_CARR21 or $TC_CARR22 contains a reference to a rotary axis in the channel, the axis position of the relevant axis at the start of the current block is used as the angle, rather than the entry in $TC_CARR13 or $TC_CARR14.

A mixed operating mode is permissible, i.e. the angles can be determined from the entry in the system variables $TC_CARR13 or $TC_CARR14 for one axis, and from the position of a channel axis for the other.

This makes it possible for machines, on which the axes used to set the toolholder with orientation capability are known within the NC, to access their position directly, whereas it was previously necessary, for example, to read system variable $AA_IM[axis] and write the result of the read operation to $TC_CARR13/14. In particular, this removes the implicit preprocessing stop when reading the axis positions.

MD20180

The rotary axis position is used with its programmed or calculated value, when the machine data:

MD20180 $MC_TOCARR_ROT_ANGLE_INCR[i] = 0 (rotary axis increment of the tool carrier that can be oriented)

If the machine data is not zero however, the position used is the nearest grid point obtained for a suitable integer value n from the equation:

φ = $MC_TOCARR_ROT_ANGLE_OFFSET[i] + n * $MC_TOCARR_ROT_ANGLE_INCR[i]

This functionality is required if the rotary axes need to be indexed and cannot, therefore, assume freely-defined positions (e.g. with Hirth tooth systems). System variable $P_TCANG[i] delivers the approximated valued and system variable $P_TCDIFF[i] the difference between the exact and the approximated value.

Frame orientation TCOFR

With `TCOFR` (determination of the angle from the orientation defined by an active frame), the increments are scaled after determination of the angle from the active frame rotation. If the requested orientation is not possible due to the machine kinematic, the machining is aborted with an alarm. This also applies if the target orientation is very close to an achievable orientation. In particular the alarm in such situations cannot be prevented through the angle approximation.

TCARR frame offset

A frame offset as a result of a toolholder change becomes effective immediately on selection of `TCARR=...`. A change in the tool length, on the other hand, only becomes effective immediately if a tool is active.

TCOFR/TCOABS frame rotation

A frame rotation does not take place on activation and a rotation which is already active is not changed. As in case T (only the tool can be rotated), the position of the rotary axes used for the calculation is dependent on the G code TCOFR/TCOABS and determined from the rotation component of an active frame or from the entries $TC_CARRn.

Activation of a frame changes the position in the workpiece coordinate system accordingly, without compensating motion by the machine itself. The ratios are shown in the figure below:

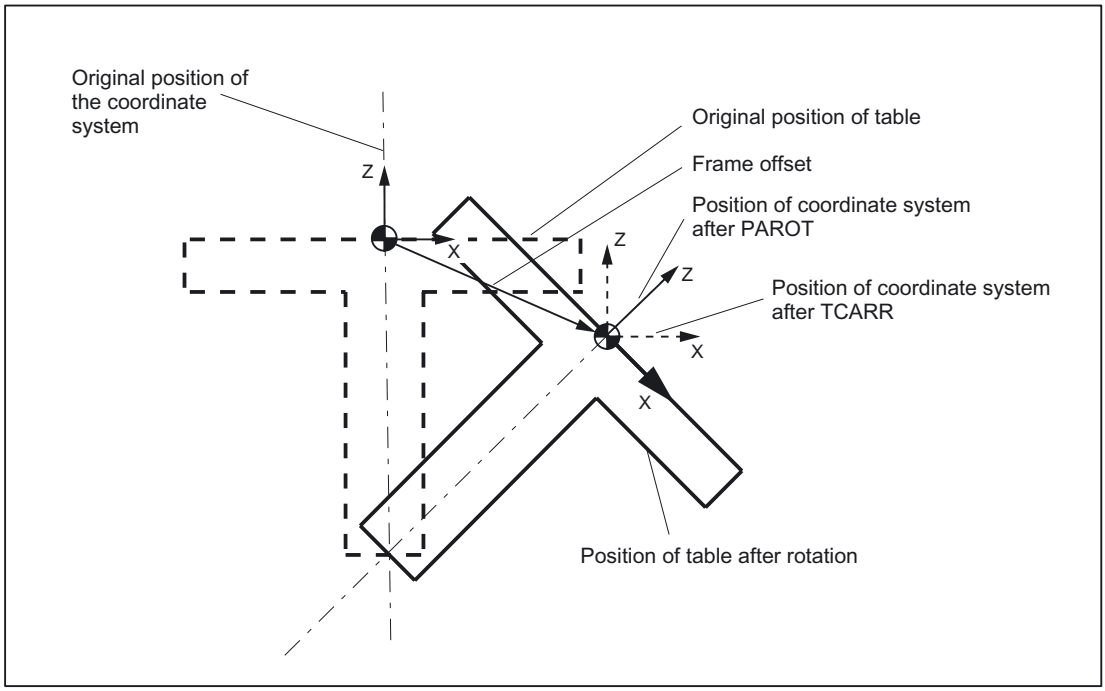


Figure 17-43 Zero offset on activation of a rotary table with TCARR

Example

On the machine in the figure, the rotary axis of the table is pointing in the positive Y direction. The table is rotated by +45 degrees. PAROT defines a frame, which similarly describes a rotation of 45 degrees about the Y axis. The coordinate system is not rotated relative to the actual environment (marked in the figure with "Position of the coordinate system after TCARR"), but is rotated by -45 degrees relative to the defined coordinate system (position after PAROT). If this coordinate system is defined with ROT Y-45, for example, and if the toolholder is then selected with active TCOFR, an angle of +45 degrees will be determined for the rotary axis of the toolholder.

Rotary table

With rotary tables (kinematic types P and M), activation with TCARR similarly does not lead to an immediate rotation of the coordinate system (see figure), i.e. even though the zero point of the coordinate system is offset relative to the machine, while remaining fixed relative to the zero point of the workpiece, the orientation remains unchanged in space.

Activation of kinematic types P and M

With kinematics of type P and M the selection of a toolholder activates an additive frame (table offset of the toolholder with orientation capability), which takes into account the zero point offset as a result of the rotation of the table.

The zero offset can be written to a dedicated system frame $P_PARTFR. For this, the bit 2 must be set in the machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK (system frames (SRAM))

The basic frame identified by following machine data is then no longer required for the zero offset:

MD20184 $MC_TOCARR_BASE_FRAME_NUMBER (number of the basic frames for taking the table offset)

Activation of kinematic type M

With kinematics of type M (tool and table are each rotary around one axis), the activation of a toolholder with TCARR simultaneously produces a corresponding change in the effective tool length (if a tool is active) and the zero offset.

Rotations

Depending on the machining task, it is necessary to take into account not only a zero offset (whether as frame or as tool length) when using a rotary toolholder or table, but also a rotation. However, the activation of a toolholder with orientation capability never leads directly to a rotation of the coordinate system.

TOROT

If only the tool can be rotated, a frame whose Z axis points in the direction of the tool can be defined with TOFRAME or TOROT.

PAROT

If the coordinate system needs to be fixed relative to the workpiece, i.e. not only offset relative to the original position but also rotated according to the rotation of the table, then PAROT can be used to activate such a rotation in a similar manner to the situation with a rotary tool.

With PAROT, the translations, scalings and mirrorings in the active frame are retained, but the rotation component is rotated by the rotation component of a toolholder with orientation capability corresponding to the table.

PAROT and TOROT take into account the overall change in orientation in cases where the table or the tool are oriented with two rotary axes. With mixed kinematics only the corresponding component caused by a rotary axis is considered. It is thus possible, for example, when using TOROT, to rotate a workpiece such that an oblique plane lies parallel to the XY plane fixed in space, whereby rotation of the tool must be taken into account in machining where any holes to be drilled, for example, are not perpendicular to this plane.

Language command PAROT is not rejected if no toolholder with orientation capability is active. This causes no changes in the programmed frame.

Note

For further information about the TCARR and TOROT as well as PAROT functions with regard to channel-specific system frames, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 721)".

17.6.5 Procedure when using toolholders with orientation capability

Creating a toolholder

The number of available toolholder data sets in the NCK is defined with machine data:

MD18088 $MN_MM_NUM_TOOL_CARRIER (maximum number of definable tool carriers)

1. The value is calculated as follows:

   MD18088 = **"Number of TO units" * "Number of toolholder data sets of a TO unit"**

   MD18088/"number of TO units" is permanently allocated to each TO unit.

   Note

   For further explanations on the definition and assignment of a TO unit by machine data:

   MD28085 $MC_MM_LINK_TOA_UNIT (assignment of a TO unit to a channel (SRAM))

   References:
   Function Manual, Extended Functions; Memory Configuration (S7)

2. Zero setting of toolholder data:

   You can use the command$TC_CARR1[0] = 0 to zero all values of all data sets.

   Individual toolholder data sets can be deleted selectively with the NC command `DELTC` or the PI service `_N_DELTCAR`.

3. Accessing the data of a toolholder:

   – Part program

     → $TC_CARRn[m] = value

     This describes the previous value of the system variables n for toolholder m with the new value "value".

     → value = $TC_CARRn[m]

     With "def real value" - the parameters of a toolholder m can be read if they have already been defined (e.g. set MD18088). Otherwise, an alarm is signaled.

   – OPI interface

   The parameters of a toolholder with orientation capability can be read and written with the NCKHMI (OPI) variable services using system variable $P_TCANG[<n>].

4. Data backup:

   The system variables specified above are saved as part of the general NCK data backup.

Selecting the toolholder

A toolholder with number m is selected with the `TCARR = m` NC program command (`TCARR`**T**ool **Carr**ier).

`TCARR = 0` deselects an active toolholder.

New tool or new toolholder

When a new tool is activated, it is always treated as if it was mounted on the active toolholder.

A new toolholder is activated immediately when it is programmed. It is not necessary to change tools or reprogram the active tool. The toolholder (number) and tool (number) are independent and can be used in any combination.

Toolholder from G code of group 42

Absolute tool orientation `TCOABS` (**T**ool **C**arrier **O**rientation **ABS**olute):

Tool orientation is determined explicitly if the corresponding values are entered in system variable $TC_CARR13 or $TC_CARR14 and G code `TCOABS` is activated in G-code group 42.

Frame tool orientation `TCOFR` (**T**ool **C**arrier **O**rientation **FR**ame):

Tool orientation can also be determined automatically from the current orientation of an active frame when selecting a tool, if one of the following G codes is active in G-code group 42 when the toolholder is selected:

- `TCOFR` or `TCOFRZ`

  The toolholder with orientation capability is set so that the tool points in the Z direction.

- `TCOFRX`

  The toolholder with orientation capability is set so that the tool points in the X direction.

- `TCOFRY`

  The toolholder with orientation capability is set so that the tool points in the Y direction.

The effect of `TCOFR` is such that, when machining on an inclined surface, tool compensations are considered implicitly as if the tool were standing vertically on the surface.

Note

The tool orientation is not bound strictly to the frame orientation. When a frame is active and G code TCOABS is active, you can select a tool, whereby the orientation of the tool is independent of the orientation of the active frame.

Following tool selection, you can change the frame, which does not affect the components of tool length compensation. It is then no longer certain that the tool is positioned perpendicular to the machining plane. You should therefore first check that the intended tool orientation is maintained on an inclined surface.

When TCOFR, etc., is active, the tool orientation used in the tool length calculation is always determined from the active frame each time the toolholder is changed.

Toolholder from G code of group 53

The G codes of group 53 (TOFRAME, TOROT, etc.) can be used to define a frame such that an axis direction (Z, Y or X) in this frame is equal to the current tool orientation.

The G code of group 6 (G17 - G19), which is active at the time TOFRAME is called, determines the tool orientation.

If no toolholder is active, or if a toolholder is active but does not cause the tool orientation to change, the Z direction in the new frame is:

- The same as the old Z direction with G17.
- The same as the old Y direction with G18.
- The same as the old X direction with G19.

These directions are modified accordingly for rotating toolholders. The same applies to the new X and Y directions.

Instead of TOFRAME or TOROT, one of the G codes TOFRAMEX, TOFRAMEY, TOROTX, or TOROTY can be used. The meanings of the axes are interchanged accordingly.

Group change

Changing the G code from group 42 (TCOABS, TCOFR, etc.) causes recalculation of the tool length components.

The (programmed) angles of rotation stored in the toolholder data are not affected, with the result that the angles originally stored in the toolholder data are reactivated on a change from TCOFR to TCOABS.

Read rotary angle (α_1 or α_2):

The angles currently used to calculate the orientation can be read via system variable $P_TCANG[n] where n = 1 or n = 2.

If two permissible solutions (i.e. a second valid pair of angles) are available for a particular orientation, the values can be accessed with $P_TCANG[3] or $P_TCANG[4]. The number of valid solutions 0 to 2 can be read with $P_TCSOL.

Tool radius compensation with CUT2D or CUT3DFS:

The current tool orientation is included in the tool radius compensation if either CUT2D or CUT3DFS is active in G-code group 22 (tool compensation type).

For nonrotating toolholders, the behavior depends solely on the active plane of G code group 6 (G17 - G19) and is, therefore, identical to the previous behavior.

All other tool compensation types:

The behavior for all other tool compensation types is unchanged.

For CUT2DF and CUT3DFF in particular, the compensation plane used for TRC is determined from the active frame, independent of the current tool orientation. Allowance is made for the active plane (G17 - G19) and the behavior is, therefore, the same as before.

The two remaining G codes of group 22, CUT3DC and CUT3DF, are not affected by the toolholder functionality because the tool orientation information in these cases is made available by the active kinematic transformation.

Two rotary axes

Two general solutions exist for two rotary axes. The control itself chooses these two solution pairs such that the orientation angles resulting from the frame are as close as possible to the specified angles.

The two following options are available for specifying the angles:

1. If $TC_CARR21 or $TC_CARR22 contains a reference to a rotary axis, the position of this axis at the start of the block in which the toolholder is activated is used to specify the angle.
2. If $TC_CARR21 or $TC_CARR22 does not contain a reference to a rotary axis, the values contained in $TC_CARR13 or $TC_CARR14 are used.

Example

The control first calculates an angle of 10 degrees for one axis. The specified angle is 750 degrees. 720 degrees (= 2 * 360 degrees) are then added to the initial angle, resulting in a final angle of 730 degrees.

Rotary axis offset

Rotary axis offsets can be specified with system variables $TC_CARR24 and $TC_CARR25. A value not equal to zero in one of these parameters means that the initial state of the associated rotary axis is the position specified by the parameter (and not position zero). All angle specifications then refer to the coordinate system displaced by this value.

When the machining plane is changed (G17 - G19), only the tool length components of the active tool are interchanged. The components of the toolholder are not interchanged. The resulting tool length vector is then rotated in accordance with the current toolholder and, if necessary, modified by the offsets belonging to the toolholder.

The two toolholder angles α_1 and α_2 are determined using a frame, independent of the active plane currently selected (G17 - G19).

Limit values

Limit angles (software limits) can be specified for each rotary axis in the system variable set ($TC_CARR30 to $TC_CARR33) used to describe the toolholder with orientation capability. These limits are not evaluated if both the minimum and maximum value is zero.

If at least one of the two limits is not equal to zero, the system checks whether the previously calculated solution is within the permissible limits. If this is not the case, an initial attempt is made to reach a valid setting by adding or subtracting multiples of 360 degrees to or from the invalid axis position. If this is impossible and two different solutions exist, the first solution is discarded and the second solution is used. The second solution is treated the same as the first with reference to the axis limits.

If the first solution is discarded and the second used instead, the contents of $P_TCANG[1/2] and $P_TCANG[3/4] are swapped, hence the solution actually used is also stored in $P_TCANG[1/2] in this case.

The axis limits are monitored even if the axis angle is specified instead of being calculated. This is the case if TCOABS is active when a toolholder with orientation capability is activated.

17.6.6 Programming

Selecting the toolholder

A toolholder is selected with the number **m** of the toolholder with:

```
TCARR = m
```

Access to toolholder data blocks

The following access is possible from the part program:

The current value of the parameter **n** for the tool holder **m** is **written** with the new "value" with::

```
$TC_CARRn[m] = value
```

The parameters of a tool holder m can, as far as the toolholder data set is already defined, **read** with:

`value = $TC_CARRn[m]` (Value must be a `REAL` variable)

The toolholder data set number must lie in the range, which is defined by the machine data:

MD18088 $MN_MM_NUM_TOOL_CARRIER (Total number of toolholder data sets that can be defined)

This number of toolholder data sets, divided by the number of active channels, can be defined for a channel.

Exception:

If settings, which deviate from the standard, are selected via the machine data:

MD28085 $MC_MM_LINK_TOA_UNIT (Assignment of TO unit to a channel).

Canceling all toolholder data blocks

All values of all toolholder data sets can be deleted from within the part program using one command.

$TC_CARR1[0] = 0

Values not set by the user are preset to 0.

Activation

A toolholder becomes active when both a **toolholder** and a **tool** have been activated. The selection of the toolholder alone has no effect. The effect of selecting a toolholder depends on the G code TCOABS / TCOFR (modal G-code group for toolholders).

Changing the G code in the TCOABS / TCOFR group causes recalculation of the tool length components when the toolholder is active. With TCOABS, the values stored in the toolholder data for both angles of rotation α1 and α2 are used to determine the tool orientation.

With TCOFR, the two angles are determined from the current frame. The values stored in the toolholder data are not changed, however. These are also used to resolve the ambiguity that can result when the angle of rotation is calculated from one frame. Here, the angle that deviates least from the programmed angle is selected from the various possible angles.

Note

For more explanations on the programming of tool compensations with toolholder kinematic and for the system variables see:

References:
Programming Manual, Job Planning

17.6.7 Supplementary conditions and control system response for orientation

Full orientation

For a given data set that describes a certain kinematic, all the conceivable special orientations can only be displayed when the following conditions are satisfied:

- The two vectors v_1 and v_2 that describe the rotary axes must also be defined (i.e. both vectors must not be equal to zero).
- The two vectors v_1 and v_2 must be perpendicular to each other.
- The tool orientation must be perpendicular to the second rotary axis.

Non-defined orientation

If these conditions are not satisfied and an orientation that cannot be achieved by an active frame is requested with TCOFR, an alarm is output.

Vector/angle of rotation dependencies

If vector v_1 or v_2, which describes the direction of a rotary axis, is set to zero, the associated angle of rotation α1 or α2 must also be set to zero. Otherwise, an alarm is produced. The alarm is not output until the toolholder is activated, i.e. when the toolholder is changed.

Tool fine compensation combined with orientation

Tool fine compensations and toolholders **cannot** be combined. The activation of tool fine compensation when a toolholder is active, and vice versa the activation of the toolholder when tool fine compensation is active, produces an alarm.

Automatic toolholder selection, RESET

For RESET or at program start, a toolholder can be selected automatically via the machine data:

MD20126 $MC_TOOL_CARRIER_RESET_VALUE (Active toolholder at RESET)

It is handled similar to the controlled selection of a tool via the machine data:

MD20120 $MC_TOOL_RESET_VALUE (Tool length compensation Power up (RESET/TP-End))

The behavior at RESET or at program start is controlled as in the case of tool selection via the same bit 6 in the machine data:

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

Or:

MD20112 $MC_START_MODE_MASK (definition of initial control system settings at NC-START)

For further information, see Section "K1: Mode group, channel, program operation, reset response (Page 489)".

SW 6.3 and higher

If TCOABS was active for the last selection before reset, the behavior is unchanged compared to previous versions. A different active G code causes the toolholder with orientation capability to be activated with the frame that was active before the last reset. Modified toolholder data ($TC_CARR...) are also considered. If these data are unchanged, the toolholder is activated in exactly the same state as before reset. If the toolholder data were changed after the toolholder selection before reset, selection corresponding to the last frame is not always possible. In this case, the toolholder with orientation capability is selected according to the G-Code (group 42) values valid at this time and the active frame.

MD22530 output of auxiliary functions to PLC

That, optionally, a constant or an M code is output when the toolholder is selected, whose number of the code is derived from the toolholder number. Can be set with the machine data:

MD22530 $MC_TOCARR_CHANGE_M_CODE (M code at toolholder change)

For further information, see Section "K1: Mode group, channel, program operation, reset response (Page 489)".

Toolholder kinematics

The following supplementary conditions must be met for toolholder kinematics:

- Tool orientation in initial state, both angles α_1 and α_2 zero, as per default setting, even if:
  - G17 parallel to Z
  - G18 Parallel to Y
  - G19 parallel to Z
- A permissible position in terms of the axis limits must be achievable.
- For any possible orientation to be set, the two rotary axes must be perpendicular to each other.

  For machines, on which the table is rotated by both axes, the tool orientation must also be perpendicular to the first rotary axis.

  For machines with mixed kinematics, the tool orientation must be perpendicular to the axis which rotates the tool, i.e. also the first rotary axis.

The following applies to orientations specified **in a frame**:

- The orientation specified in a frame must be achievable with the defined toolholder kinematics, otherwise an alarm is output.

  This situation can occur if the two rotary axes required to define the kinematics are not perpendicular to each other.

  This applies if fewer than two rotary axes are defined and is the case:
  - With **kinematic type T with rotary tool**, if the tool axis, which defines the tool direction, is not perpendicular to the **second** axis.
  - With **kinematic types M and P with rotary workpiece**, if the tool axis, which defines the tool direction, is not perpendicular to the **first** axis.
- Rotary axes, which require a frame with a defined tool orientation in order to reach a specific position, are only determined unambiguously in the case of one rotary axis. Two general solutions exist for two rotary axes.
- In all cases where ambiguities may arise, it is particularly important that the approximate angles expected from the frame are stored in the tool data, and that the rotary axes are in the vicinity of the expected positions.

Response with ASUP, REPOS

The toolholder can be changed in an asynchronous subprogram (ASUB). When the interrupted program is resumed with REPOS, the approach motion of the new toolholder is taken into account and the program continues with this motion. The treatment here is analogous to tool change in an ASUB.

For further information, see Section "K1: Mode group, channel, program operation, reset response (Page 489)".

17.7 Cutting edge data modification for tools that can be rotated

17.7.1 Function

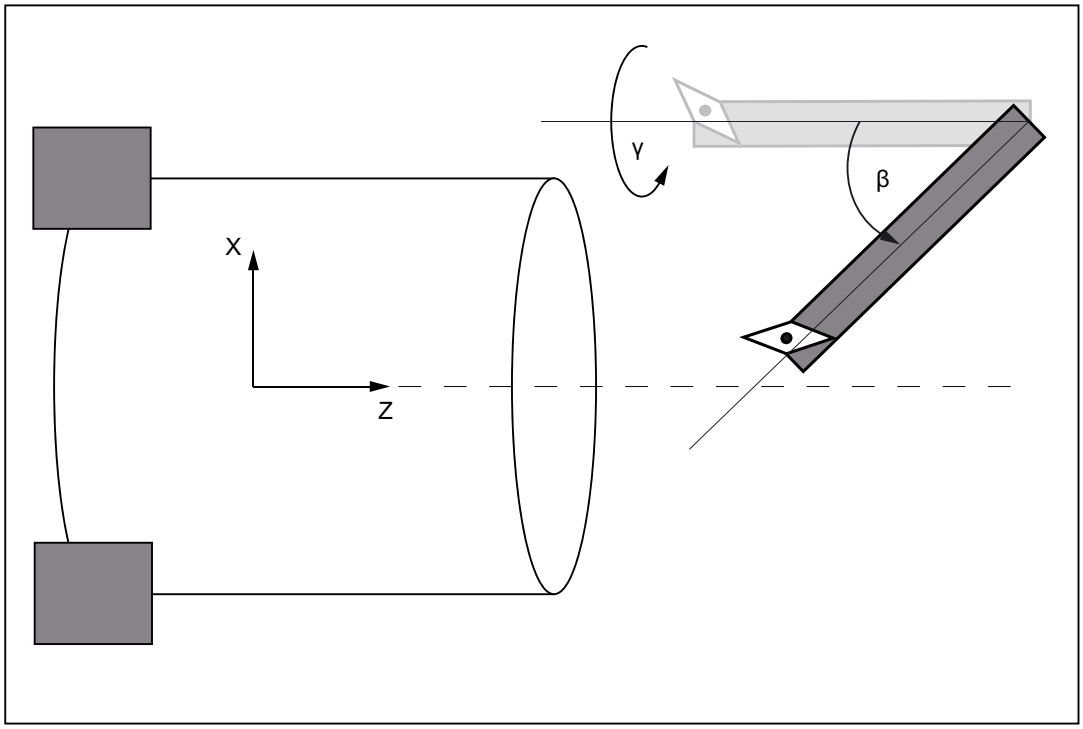
Using the function "cutting data modification for rotatable tools", the changed geometrical relationships, that are obtained relative to the workpiece being machined when rotating tools (predominantly turning tools, but also drilling and milling tools) can be taken into account.

17.7.2 Determination of angle of rotation

The current rotation of the tool is always determined from a currently active, orientable toolholder (see Section "Toolholder with orientation capability (Page 1657)").

The angle of rotation of the toolholder with orientation capability is normally (but not necessarily) defined with the`TCOFR` command from an active frame. This method can be used to define the tool orientation independently of the actual kinematics with which the tool is rotated, identically in each case with the help of two angles.

The two machine-independent orientation angles β (Beta) and γ (Gamma) are used to define the tool rotation. β is the angle of rotation and the applicate (typically a B axis in `G18`) and γ a rotation around the ordinate (Typically a C axis in `G18`). The rotation is first executed around Y, finally around β, i.e. the y axis is rotated by the β axis:



17.7.3 Cutting edge position, cut direction and angle for rotary tools

Turning tools

Turning tools means the following tools whose tool type ($TC_DP1) has values in the range of 500 to 599. Grinding tools (tool types 400 to 499) are equivalent to turning tools.

Tools are treated independently of tool type such as turning tools if:

SD42950 $SC_TOOL_LENGTH_TYPE = 2

Cutting edge position and cut direction

Turning tools are limited by their main and secondary cutting edges. The tool parameter "Cutting edge position" is defined via the position of these two cutting edges relative to the coordinate axes. The ratios are displayed with diagram in the following figure:

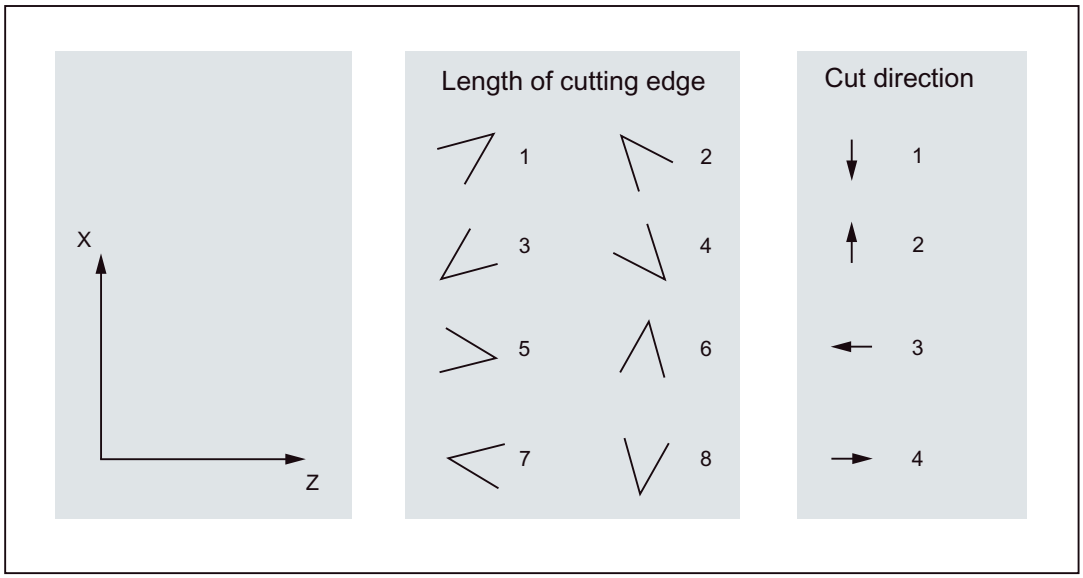


Figure 17-44 Cutting edge position and cut direction for turning tools

The values 1 to 4 characterize the cases in which both cutting edges lie in the same quadrant; the values 5 to 8 characterize the cases in which both cutting edges lie in neighboring quadrants or there is a coordinate axis between the two cutting edges. The cutting edge position is stored in the tool parameter $TC_DP2.

A cut direction can be defined for each turning tool. It is stored in the tool parameter $TC_DP11. It has values between 1 and 4, and it characterizes a positive or negative direction of the coordinate axes:

Value:	Meaning:
1	Ordinate **-**
2	Ordinate **+**
3	Abscissa **-**
4	Abscissa **+**

Two different cut directions can be assigned to each cutting edge position:

Cutting edge position:	1	2	3	4	5	6	7	8
Cut direction:	2, 4	2, 3	1, 3	1, 4	1, 2	3, 4	1, 2	3, 4

Holder angle and clearance angle

The following figure depicts the two angles (holder angle and clearance angle) of a turning tool with cutting edge position 3, that are necessary for describing the geometry of the tool cutting edge. The cut direction in this example is 3, i.e. it denotes the negative Z direction (abscissa direction for `G18`).

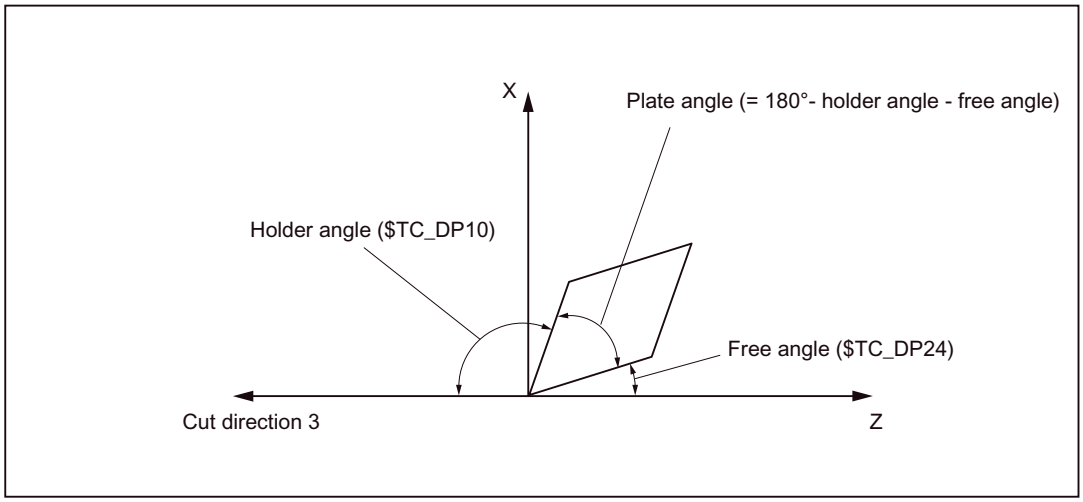


Figure 17-45 Angle and cut direction for a turning tool with cutting edge position 3

The cut direction specifies the reference direction of the holder angle. The clearance angle is the angle measured between the inverse cut direction and the adjacent cutting edge (positive). Holder angle and clearance angle are stored in the tool parameters $TC_DP10 or $TC_DP24.

Note

Cut direction and tool angle are relevant only in the cutting edge positions 1 to 8.

17.7.4 Modifications during the rotation of turning tools

Tool orientation

Unlike milling tools, turning tools are not rotation-symmetric. This means that normally 3 degrees of freedom or three rotary axes are required to describe the tool orientation. The concrete kinematics therefore, is independent of the machine only to the extent the desired orientation can be set. If necessary, the third degree of freedom can be substituted by a rotation of the tool coordinate system.

Note

The division of the orientation into one component created by the toolholder with orientation capability and a second componnet achieved via a rotation of the coordinate system is the responsibility of the application. The control does not provide any further functionality in this regard.

Shape of cutting edge

If a turning tool turns by an angle against the machining plane (i.e. around an axis in the machining plane, typically a C axis) that is not a multiple of 180°, then the configuration of the (circular) tool cutting edge in the machining plane becomes an ellipse. It is assumed that the deviations from the circular form arising on account of such rotations is so insignificant that they can be ignored (tilt angle < 5°), i.e. the control always ignores the tool orientation and assumes a circular cutting edge.

This also means that with reference to the active plane, the control accepts only a rotation by 180° as a setting deviating from the initial position. This limitation is valid for the shape of cutting edge only. The tool lengths are always considered correctly in random spatial rotations.

Cutting edge position, cut direction and angle

A rotation by 180° around an axis in the machining plane means that while using the tool at the same position, the spindle rotation direction with reference to the use of the unturned tool must be inverted.

Cut direction and cutting edge position are also not modified like the cutting edge reference point (see below) if the tool is rotated from the plane by +/- 90° (with a tolerance of app. 1°) because then the configuration of the cutting edge is not defined in the current plane.

If the tool rotates in the plane (rotation around an axis vertical to the machining place or around the Y axis for `G18`), the cutting edge position is determined from the resulting angle for the clearance and holder angles. If these two angles are not specified for the tool (i.e. $TC_DP10 and $TC_DP24 are both zero), then the new cutting edge position is determined from the turning angle alone. The cutting edge position changes only in 90° steps, i.e. the cutting edge position remains independent of the initial state either in the value range 1 to 4 or 5 to 8. The new cutting edge position is then determined exclusively from the angle of rotation if the specified values for holder angle and clearance angle are not allowed (negative values, resulting plate angle negative or more than 90°). Clearance angle and holder angle are not modified in all these cases.

Depending on the rotation, the cut direction is modified in such a way that the resulting clearance angle remains less than 90°. If the original cut direction and the original cutting edge position do not match, then the cut direction is not modified during rotation of the tool (see Section "Cutting edge position, cut direction and angle for rotary tools (Page 1690)").

The angle of rotation in the plane, as it was determined from the toolholder with orientation capability, is available in the system variable $P_CUTMOD_ANG or $AC_CUTMOD_ANG. This angle is the original angle without any final rounding to multiples of 45° or 90°.

Limit cases

If, for a turning tool, the cutting edge position, cut direction , clearance and holder angles have valid values so that all cutting edge positions (1 to 8) are possible through suitable rotations in the plane, then the cutting edge positions 1 to 4 are preferred to cutting edge positions 5 to 8 in the cases in which one of the cutting edges (main or secondary cutting edge) is away from the coordinate axis by less than half the input increment ((0.0005° for an input specification of 3 decimal digits).

The following is applicable in all other cases (milling tools or turning tools without valid cutting edge parameters) in which rotation is possible only in 90° steps: If the amount of the rotation angle is smaller than 45° + 0.5 input increments (corresponds to 45.0005° for an input specification of three decimal places), the cutting edge position and cut direction are not changed, i.e. these cases are treated as rotations that are smaller than 45°. Rotations, the amount of which deviates from 180° by less than 45° + 0.5 input increments are treated identically as rotations in the range of 135° to 225°.

Cutting edge reference point

The cutting edge center point and the cutting edge reference point are defined for turning tools. The position of these two points relative to each other is defined by the cutting edge position.

The distance of the two points for cutting edge positions 1 to 4 is equal to √2 times the cutting edge radius; for cutting edge positions 5 to 8 it is equal to 1 times the cutting edge radius. In the first case, the cutting edge reference point relative to the cutting edge center point lies in the machining plane on a bisecting line, while in the second case it lies on a coordinate axis.

If your rotate the tool by a random angle around an axis vertical to the machining plane, the the cutting edge reference point would also rotate if it had a fixed position relative to the tool. The above-mentioned condition (position on an axis or a bisecting axis) is not fulfilled in most cases. This is not desirable. Instead, the cutting edge reference point should always be modified in such a way that the distance vector between cutting edge reference point and cutting edge center point has one of the mentioned 8 directions. The cutting edge position must be modified for this if necessary.

The ratios are shown with examples in the figure below:

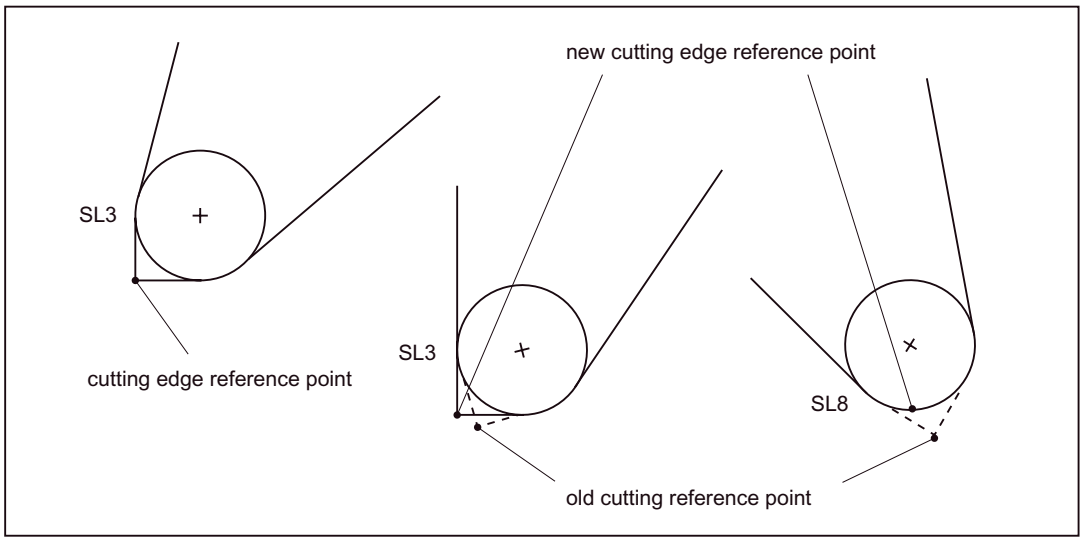


Figure 17-46 Cutting edge reference point and cutting edge position (SL) for tool rotation

A tool with the cutting edge position 3, the clearance angle 22.5° and holder angle 112.5° is rotated. For rotations up to 22.5°, the cutting edge position is maintained, the position of the cutting edge reference point relative to the tool however, is compensated in such a way that the relative position of both points are maintained in the machining plane. For bigger rotations (up to 67.5°), the cutting edge position changes to value 8.

Note

As the cutting edge reference point is defined by the tool length vector, modifying the cutting edge reference point changes the effective tool length.

17.7.5 Cutting edge position for milling and tapping tools

Milling and tapping tools

Milling and tapping tools means the following tools whose tool type ($TC_DP1) has values in the range of 100 to 299.

Tools are treated independently of tool type such as milling and tapping tools if:

SD42950 $SC_TOOL_LENGTH_TYPE = 1

Length of cutting edge

A cutting edge position is also introduced for the so defined milling and tapping tools which is modified according to the following description, in case of rotations.

Any specified cutting edge position for tools that are not milling and tapping tools or turning tools according to the mentioned definitions, is not evaluated.

The cutting edge position of the tapping and milling tools is stored in tool parameter $TC_DP2 as in the case of turning tools. Based on the definition of the cutting edge position for turning tools, this parameter can assume the values 5 to 8. Here, the cutting edge position specifies the orientation (the direction of the rotation axis) of the tool:

Length of cutting edge	Direction of rotation axis of tool
5	Abscissa **+**
6	Ordinate **+**
7	Abscissa **-**
8	Ordinate **-**

Example:

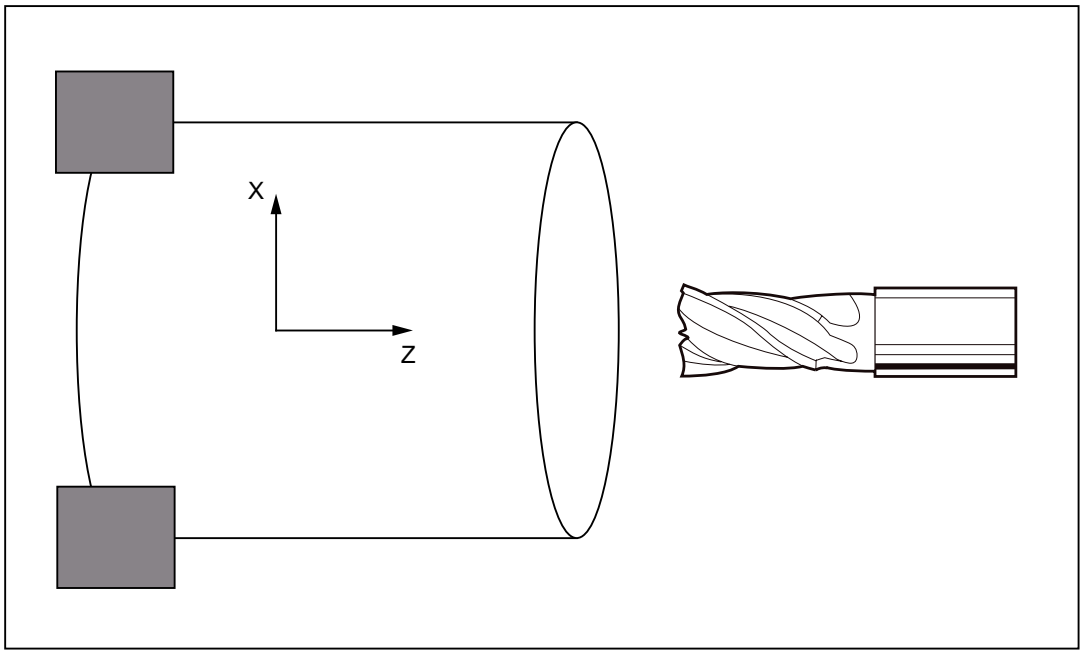


Figure 17-47 Milling tool with cutting edge position 7

17.7.6 Modifications during rotation of milling and tapping tools

The cutting edge position is recalculated appropriately during a rotation of a milling or tapping tool. Cut direction and tool angle (clearance angle or holder angle) are not defined for milling and tapping tools so that the change in cutting edge position is derived exclusively from the rotation. Thus, for milling and tapping tools, the cutting edge position always changes when the amount of rotation with reference to the zero setting is more than 45°.

17.7.7 Parameter assignment

Reaction to errors

Different fault conditions can occur during the activation of the "Cutting edge data modification for rotary tools" function (via explicit call with CUTMOD or through a tool selection).

For each of these possible fault conditions, one can define whether the error is to trigger an alarm output, whether such an alarm is only to be displayed (warning), or whether additionally the interpretation of the part program must be cancelled. The setting is done via the following machine data:

MD20125 $MC_CUTMOD_ERR

Two bits of the machine data are assigned to each fault condition:

Fault condition	Bit	Description
No valid cut direction is defined for the active tool.	0	Alarm output for error "Invalid cut direction"
	1	Program stop for error "Invalid cut direction"
The cutting edge angle (clearance angle and holder angle) of the active tool are both zero.	2	Alarm output for error "Not defined cutting edge angle"
	3	Program stop for error "Not defined cutting edge angle"
The clearance angle of the active tool has an impermissible value (< 0° or > 180°).	4	Alarm output for error "Invalid clearance angle"
	5	Program stop for error "Invalid clearance angle"
The holder angle of the active tool has an impermissible value (< 0° or > 90°).	6	Alarm output for error "Invalid holder angle"
	7	Program stop for error "Invalid holder angle"
The plate angle of the active tool has an impermissible value (< 0° or > 90°).	8	Alarm output for error "Invalid plate angle"
	9	Program stop for error "Invalid plate angle"
The cutting edge position - holder angle combination of the active tool is not permitted (the holder angle must be ≤ 90° for cutting edge position 1 to 4; for cutting edge positions 5 to 8 it must be ≥ 90°).	10	Alarm output for error "Invalid cutting edge position - holder angle combination"
	11	Program stop for error "Invalid cutting edge position - holder angle combination"
Inadmissible rotation of the active tool (the tool was rotated from the active machining plane by ± 90° (with a tolerance of about 1°). Hence the cutting edge position is no longer defined in the machining plane.	12	Alarm output for error "Invalid rotation"
	13	Program stop for error "Invalid rotation"

Response to POWER ON

The "cutting edge data modification for rotary tools" function (CUTMOD) is initialized automatically during POWER ON with the value stored in machine data:

MD20127 $MC_CUTMOD_INIT

If the value of this machine data is "-2", CUTMOD is set to the value that is set in machine data:

MD20126 $MC_TOOL_CARRIER_RESET_VALUE (Active toolholder at RESET)

17.7.8 Programming

The "Cutting edge data modification for rotary tools" function is activated with the `CUTMOD` command.

Syntax

```
CUTMOD=<value>
```

Meaning

`CUTMOD` Command to switch-in the function "cutting data modification for tools that can be rotated"

`<value>` The following values can be assigned to the `CUTMOD` command:

0 The function is deactivated.

The values supplied from system variables $P_AD... are the same as the corresponding tool parameters.

> 0 The function is activated if a toolholder that can be orientated with the specified number is active, i.e. the activation is linked to a specific toolholder that can be orientated.

The values supplied from system variables $P_AD... may be modified with respect to the corresponding tool parameters depending on the active rotation.

The deactivation of the designated toolholder that can be orientated temporarily deactivates the function; the activation of another toolholder that can be orientated permanently deactivates it. This is the reason that in the first case, the function is re-activated when again selecting the same toolholder that can be orientated; in the second case, a new selection is required - even if at a subsequent time, the toolholder that can be orientated is re-activated with the specified number.

The function is not influenced by a reset.

-1 The function is always activated if a toolholder that can be orientated is active.

When changing the toolholder or when de-selecting it and a subsequent new selection, `CUTMOD` does not have to be set again.

-2 The function is always activated if a toolholder that can be orientated is active whose number is the same as the currently active toolholder that can be orientated.

If a toolholder that can be orientated is not active, then this has the same significance as CUTMOD=0.

If a toolholder that can be orientated is active, then this has the same significance as when directly specifying the actual toolholder number.

< -2 Values less than 2 are ignored, i.e. this case is treated as if CUTMOD was not programmed.

Note:
This value range should not be used as it is reserved for possible subsequent expansions.

Note

SD42984 $SC_CUTDIRMOD

The function that can be activated using the CUTMOD command replaces the function that can be activated using the setting data SD42984 $SC_CUTDIRMOD. However, this function remains available unchanged, because it doesn't make sense to use both functions in parallel, it can only be activated if CUTMOD is equal to zero.

Effectiveness of the modified cutting data

The modified tool nose position and the modified tool nose reference point are immediately effective when programming, even for a tool that is already active. A tool does not have to be re-selected for this purpose.

Influence of the active machining plane

To determine modified tool nose position, cutting direction and holder or clearance angle, the evaluation of the cutting edge in the active plane (G17 - G19) is decisive.

However, if setting data SD42940 $SC_TOOL_LENGTH_CONST (change of the tool length component when selecting the plane), contains a valid value not equal to zero (plus or minus 17, 18 or 19), then its contents define the plane in which the relevant quantities are evaluated.

System variables

The following system variables are available:

System variables	Meaning
$P_CUTMOD_ANG / $AC_CUTMOD_ANG	Supplies the (non-rounded) angle in the active machining plane that was used as basis for the modification of the cutting data (tool nose position, cut direction, clearance angle and holder angle) for the functions activated using CUTMOD and/or $SC_CUTDIRMOD. $P_CUTMOD_ANG refers to the actual state in the preprocessing, $AC_CUTMOD_ANG to the actual main run block.
$P_CUTMOD / $AC_CUTMOD	Reads the currently valid value that was last programmed using the command CUTMOD (number of the toolholder that should be activated for the cutting data modification). If the last programmed CUTMOD value = -2 (activation with the currently active toolholder that can be orientated), then the value -2 is not returned in $P_CUTMOD, but the number of the active toolholder that can be orientated at the time of programming. $P_CUTMOD refers to the actual state in the preprocessing, $AC_CUTMOD to the actual main run block.
$P_CUT_INV / $AC_CUT_INV	Supplies the value TRUE if the tool is rotated so that the spindle direction of rotation must be inverted. To do this, the following four conditions must be fulfilled in the block to which the read operations refer: 1. If a turning or grinding tool is active (tool types 400 to 599 and / or SD42950 $SC_TOOL_LENGTH_TYPE = 2). 2. If the cutting influence was activated using the language command CUTMOD. 3. If a toolholder that can be orientated is active, which was designated using the numerical value of CUTMOD. 4. If the toolholder that can be orientated rotates the tool around an axis in the machining plane (this is typically the C axis) so that the resulting perpendicular of the tool cutting edge is rotated with respect to the initial position by more than 90° (typically 180°). The contents of the variable is FALSE if at least one of the specified four conditions is not fulfilled. For tools whose tool nose position is not defined, the value of the variable is always FALSE. $P_CUT_INV refers to the actual state in the preprocessing and $AC_CUT_INV to the actual main run block.

All main run variables ($AC_CUTMOD_ANG, $AC_CUTMOD and $AC_CUT_INV) can be read in synchronized actions. A read access operation from the preprocessing generates a preprocessing stop.

Modified cutting data

If a tool rotation is active, the modified data is made available in the following system variables:

System variable	Meaning
$P_AD[2]	Cutting edge position
$P_AD[10]	Holder angle
$P_AD[11]	Cut direction
$P_AD[24]	Clearance angle

Note

The data is always modified with respect to the corresponding tool parameters ($TC_DP2[..., ...] etc.) if the function "cutting data modification for rotatable tools" was activated using the command CUTMOD and a toolholder that can be orientated, which causes a rotation, is activated.

17.7.9 Example

The following example refers to a tool with tool nose position 3 and a toolholder that can be orientated, which can rotate the tool around the B axis.

The numerical values in the comments specify the end of block positions in the machine coordinates (MCS) in the sequence X, Y, Z.

```
Program code                                                    Comment
N10 $TC_DP1[1,1]=500
N20 $TC_DP2[1,1]=3                                              ; Length of cutting edge
N30 $TC_DP3[1,1]=12
N40 $TC_DP4[1,1]=1
N50 $TC_DP6[1,1]=6
N60 $TC_DP10[1,1]=110                                           ; Holder angle
N70 $TC_DP11[1,1]=3                                             ; Cut direction
N80 $TC_DP24[1,1]=25                                            ; Clearance angle

N90 $TC_CARR7[2]=0 $TC_CARR8[2]=1 $TC_CARR9[2]=0                ; B axis
N100 $TC_CARR10[2]=0 $TC_CARR11[2]=0 $TC_CARR12[2]=1            ; C axis
N110 $TC_CARR13[2]=0
N120 $TC_CARR14[2]=0
N130 $TC_CARR21[2]=X
N140 $TC_CARR22[2]=X
N150 $TC_CARR23[2]="M"
```

```
Program code                                    Comment
N160 TCOABS CUTMOD=0
N170 G18 T1 D1 TCARR=2                            X          Y          Z
N180 X0 Y0 Z0 F10000                            ; 12.000     0.000      1.000

N190 $TC_CARR13[2]=30
N200 TCARR=2
N210 X0 Y0 Z0                                   ; 10.892     0.000      -5.134
N220 G42 Z-10                                   ; 8.696      0.000      -17.330
N230 Z-20                                       ; 8.696      0.000      -21.330
N240 X10                                        ; 12.696     0.000      -21.330
N250 G40 X20 Z0                                 ; 30.892     0.000      -5.134

N260 CUTMOD=2 X0 Y0 Z0                          ; 8.696      0.000      -7.330
N270 G42 Z-10                                   ; 8.696      0.000      -17.330
N280 Z-20                                       ; 8.696      0.000      -21.330
N290 X10                                        ; 12.696     0.000      -21.330
N300 G40 X20 Z0                                 ; 28.696     0.000      -7.330

N310 M30
```

Explanations:

In block N180, initially the tool is selected for CUTMOD=0 and non-rotated toolholders that can be orientated. As all offset vectors of the toolholder that can be orientated are 0, the position that corresponds to the tool lengths specified in $TC_DP3[1,1] and $TC_DP4[1,1] is approached.

The toolholder that can be orientated with a rotation of 30° around the B axis is activated in block N200. As the tool nose position is not modified due to CUTMOD=0, the old tool nose reference point is decisive just as before. This is the reason that in block N210 the position is approached, which keeps the old tool nose reference point at the zero (i.e. the vector (1, 12) is rotated through 30° in the Z/X plane).

In block N260, contrary to block N200 CUTMOD=2 is effective. As a result of the rotation of the toolholder that can be orientated, the modified tool nose position becomes 8. The consequence of this is also the different axis positions.

The tool radius compensation (TRC) is activated in blocks N220 and/or N270. The different tool nose positions in both program sections has no effect on the end positions of the blocks in which the TRC is active; the corresponding positions are therefore identical. The different tool nose positions only become effective again in the deselect blocks N260 and/or N300.

17.8 Incrementally programmed compensation values

17.8.1 G91 extension

Requirements

Incremental programming with `G91` is defined such that the compensation value is traversed additively to the incrementally programmed value when a tool compensation is selected.

Applications

For applications such as scratching, it is necessary only to traverse the path programmed in the incremental coordinates. The activated tool compensation is not traversed.

Sequence

Selection of a tool compensation with incremental programming

- Scratch workpiece with tool tip.
- Save the actual position in the basic frame (set actual value) after reducing it by the tool compensation.
- Traverse incrementally from the zero position.

Activation

It is possible to set whether a changed tool length is traversed with `FRAME` and incremental programming of an axis, or whether only the programmed path is traversed with the setting data:

SD42442 $SC_TOOL_OFFSET_INCR_PROG (tool length compensations)

Zero offset / frames G91

It is possible to set whether a zero offset is traversed as standard with value = 1 with FRAME and incremental programming of an axis, or whether only the programmed path is traversed with value = 0 with the setting data:

SD42440 $SC_FRAME_OFFSET_INCR_PROG (zero offset in frames)

For further information, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 721)".

Supplementary condition

If the behavior is set such that the offset remains active even after the end of the program and RESET
MD20110 $MC_RESET_MODE_MASK, bit6=1 (specification of the controller initial setting after reset / TP end)
and if an incremental path is programmed in the first part program block, the compensation is always traversed additively to the programmed path.

Note

With this configuration, part programs must always begin with absolute programming.

17.8.2 Machining in direction of tool orientation

Typical application

On machines with toolholders with orientation capability, traversing should take place in the tool direction (typically, when drilling) without activating a frame (e.g., using TOFRAME or TOROT), on which one of the axes points in the direction of the tool.

This is also true of machines on which a frame defining the oblique plane is active during oblique machining operations, but the tool cannot be set exactly perpendicular because an indexed toolholder (Hirth tooth system) is restricting the setting of the tool orientation.

In these cases it is then necessary - contrary to the motion actually requested perpendicular to the plane - to drill in the tool direction, as the drill would otherwise not be guided in the direction of its longitudinal axis, which, among other things, would lead to breaking of the drill.

MOVT

The end point of such a motion is programmed with MOVT= .... The programmed value is effective incrementally in the tool direction as standard. The positive direction is defined from the tool tip to the toolholder. The content of MOVT is thus generally negative for the infeed motion (when drilling), and positive for the retraction motion. This corresponds to the situation with normal paraxial machining, e.g., with G91Z ....

If the motion is programmed in the form MOVT=AC(...), MOVT functions absolutely. In this case a plane is defined, which runs through the current zero point, and whose surface normal vector is parallel to the tool orientation. MOVT then gives the position relative to this plane:

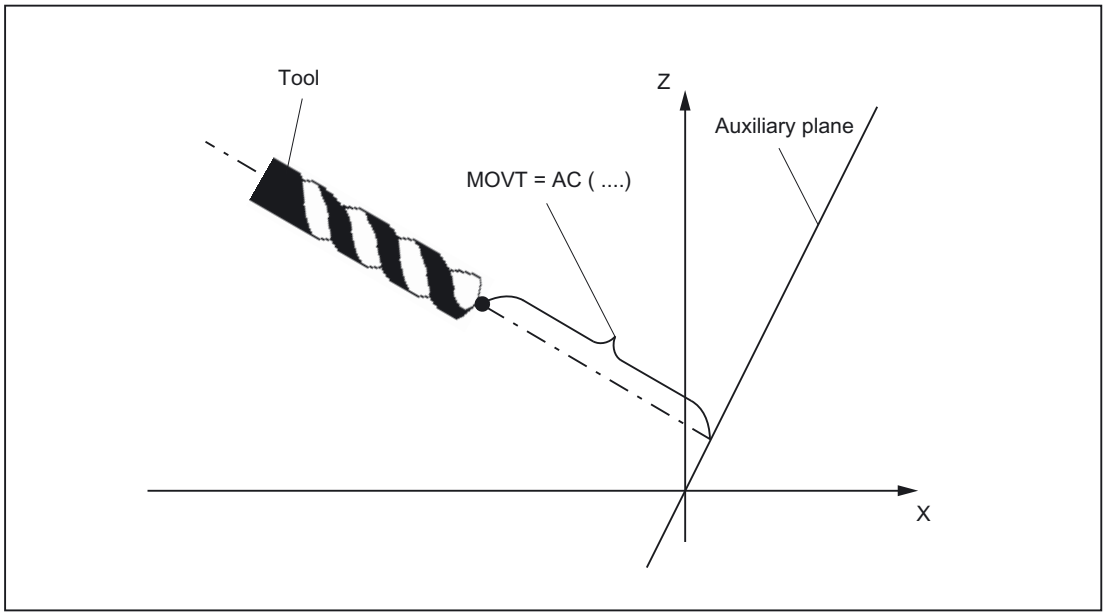


Figure 17-48 Definition of the position for absolute programming of a motion in tool direction

The reference to this auxiliary plane serves only to calculate the end position. Active frames are not affected by this internal calculation.

Instead of MOVT= ... it is also possible to write MOVT=IC(...) if it is to be plainly visible that MOVT is to function incrementally. There is no functional difference between the two forms.

Supplementary conditions

The following supplementary conditions apply to programming with MOVT:

- It is independent of the existence of a toolholder with orientation capability. The direction of the motion is dependent on the active plane. It runs in the direction of the vertical axes, i.e., with G17 in Z direction, with G18 in Y direction and with G19 in X direction. This applies both where no toolholder with orientation capability is active and for the case of a toolholder with orientation capability without rotary tool or with a rotary tool in its basic setting.
- MOVT acts similarly for active orientation transformation (345axis transformation).
- If in a block with MOVT the tool orientation is changed simultaneously (e.g., active 5axis transformation by means of simultaneous interpolation of the rotary axes), the orientation at the start of the block is decisive for the direction of movement of MOVT. The path of the tool tip (TCP - Tool Center Point) is not affected by the change in orientation.
- Linear or spline interpolation (G0, G1, ASPLINE, BSPLINE, CSPLINE) must be active. Otherwise, an alarm is produced. If a spline interpolation is active, the resultant path is generally not a straight line, since the end point determined by MOVT is treated as if it had been programmed explicitly with X, Y, Z.
- A block with MOVT must not contain any programming of geometry axes (alarm 14157).

17.9 Basic tool orientation

Application

Normally, the orientation assigned to the tool itself depends exclusively on the active machining plane. For example, the tool orientation is parallel to Z with G17, parallel to Y with G18 and parallel to X with G19.

Different tool orientations can only be programmed by activating a 5axis transformation. The following system variables have been introduced in order to assign a separate orientation to each tool cutting edge:

System variable	Description of tool orientation	Format	Preassignment
$TC_DPV[t, d]	Tool cutting edge orientation	INT	0
$TC_DPV3[t, d]	L1 component of tool orientation	REAL	0
$TC_DPV4[t, d]	L2 component of tool orientation	REAL	0
$TC_DPV5[t, d]	L3 component of tool orientation	REAL	0

Indexing: Same as tool system variable $TC_DPx[t, d]
t: T number of cutting edge
d: D number of cutting edge

Identifiers $TC_DPV3 to $TC_DPV5 are analogous to identifiers $TC_DP3 to $TC_DP5 of the tool length components.

MD18114

The system variables for describing the tool orientation are only available if machine data is not equal to zero:

MD18114 $MN_MM_ENABLE_TOOL_ORIENT (assign orientation to tool cutting)

MD18114 $MN_MM_ENABLE_TOOL_ORIENT	
Value = 1	Only system variable $TC_DPV[t, d] is available.
Value = 2	All four system variables are available.

Define direction vector

If all four system variables contain 0, the orientation is defined only by the active plane (as before).

If system variable $TC_DPV[t, d] is equal to zero, the other three parameters - if available - define a direction vector. The amount of the vector is insignificant.

Example:

```
$TC_DPV[1, 1] = 0
$TC_DPV3[1, 1] = 1.0
$TC_DPV4[1, 1] = 0.0
$TC_DPV5[1, 1] = 1.0
```

In this example, the basic orientation points in the direction of the bisector in the L1L3 plane, i.e., the basic orientation in the bisector for a milling tool and active plane `G17` lies in the Z/X plane.

Basic orientation of tools

Basic orientation of:	With :
Turning and grinding tools	`G18`
Milling tools	`G17`

The active tool orientation is unchanged in these cases and is equivalent to the original settings in $TC_DPVx[t, d].

The basic orientation is always the direction perpendicular to the plane in which tool radius compensation is performed. With turning tools, in particular, the tool orientation generally coincides with the longitudinal tool axis.

The setting data specified below are effective only if the basic orientation of the tool is defined by an entry in at least one of the system variables $TC_DPVx[t, d].

They have no effect if the tool orientation is only determined by the plane selection `G17` - `G19` and is compatible with previous behavior.

The plane of the basic orientation for a cutting edge is treated either like a milling tool or like a turning tool, irrespective of the entry in $TC_DP1, if the following setting data is not equal to zero:

SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type)

Plane change

A change of plane causes a change in orientation.

The following rotations are initiated:

When changing from:	Rotations
G17 ⇒ G18: G18 ⇒ G19: G19 ⇒ G17:	Rotation through -90 degrees about the Z axis followed by rotation through -90 degrees about the X axis
G17 ⇒ G19: G18 ⇒ G17: G19 ⇒ G18:	Rotation through 90 degrees about the X axis followed by rotation through 90 degrees about the Z axis

These rotations are the same as those that have to be performed in order to interchange the components of the tool length vector on a change of plane.

The basic orientation is also rotated when an adapter transformation is active.

If the following setting data is not equal to zero, the tool orientation is not rotated on a change of plane:

SD42940 $SC_TOOL_LENGTH_CONST (change of tool length components on change of planes).

Tool length components

The components of the tool orientation are treated the same as the components of the tool length, with respect to setting data:

SD42910 $SC_MIRROR_TOOL_LENGTH (Sign change tool wear when mirroring).

SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type)

Therefore the components are changed respectively and assigned to the geometry axis.

System variable $TC_DPV[t, d]

The purpose of system variable $TC_DPV[t, d] is to allow the simple specification of certain basic orientations (parallel to coordinate axes) that are required frequently. The permissible values are shown in the table below. The values in the first and second/third columns are equivalent.

$TC_DPV[t, d]	Basic orientation	
	Milling tools *	Turning tools *
≤ 0 or > 6	($TC_DPV5[t, d], $TC_DPV4[t, d], $TC_DPV3[t, d],) **	($TC_DPV3[t, d], $TC_DPV5[t, d], $TC_DPV4[t, d],) **
1	(0, 0, V)	(0, V, 0)
2	(0, V, 0)	(0, 0, V)
3	(V, 0, 0)	(V, 0, 0)
4	(0, 0, -V)	(0, -V, 0)
5	(0, -V, 0)	(0, 0, -V)
6	(-V, 0, 0)	(-V, 0, 0)

* Turning tools in this context are any tools whose tool type ($TC_DP1[t, d]) is between 400 and 599. All other tool types refer to milling tools.

** If all three values $TC_DPV3[t, d], $TC_DPV4[t, d], $TC_DPV5[t, d] are equal to zero in this case, the tool orientation is determined by the active machining plane (default).

V Stands for a positive value in the corresponding system variables.

Example:

For milling tools:

$TC_DPV[t, d] = 2 is equal to:

$TC_DPV3[t, d] = 0, $TC_DPV4[t, d] = 0, $TC_DPV5[t, d] = V.

Supplementary conditions

If the "Scratch" function is used in the RESET state, the following must be noted with respect to the initial setting:

- The wear components are evaluated depending on the initial settings of the G-code groups TOWSTD, TOWMCS and TOWWCS.
- If a value other than the initial setting is needed to ensure correct calculation, scratching may be performed only in the STOP state.

Note

"Special handling of tool compensations" pays particular attention to tool compensations with evaluation of sign for tool length with wear and temperature fluctuations.

The following are taken into account:

- Tool type
- Transformations for tool components
- Assignment of tool length components to geometry axes independently of tool type

17.10 Special handling of tool compensations

17.10.1 Relevant setting data

SD42900- 42960

Setting data SD42900 - SD42940 can be used to make the following settings with reference to tool compensation:

- Sign of the tool length
- Sign of the wear
- Behavior of the wear components when mirroring geometry axes
- Behavior of the wear components when changing the machining plane via setting data
- Allocation of the tool length components independent of actual tool type
- Transformation of wear components into a suitable coordinate system for controlling the effective tool length

Note

In the following description, the wear includes the total values of the following components:

- Wear values: $TC_DP12 to $TC_DP20
- Sum offset, consisting of:
  - Wear values: $SCPX3 to $SCPX11
  - Setup values: $ECPX3 to $ECPX11

You will find detailed information about sum and tool offsets in:

References:
Function Manual Tool Management

Programming Manual. Fundamentals; Tool Offsets

Required setting data

- SD42900 $SC_MIRROR_TOOL_LENGTH (mirroring of tool length components and components of the tool base dimension)
- SD42910 $SC_MIRROR_TOOL_WEAR (mirroring of wear values of tool length components)
- SD42920 $SC_WEAR_SIGN_CUTPOS (sign evaluation of the wear components)
- SD42930 $SC_WEAR_SIGN (inverts the sign of the wear dimensions)
- SD42940 $SC_TOOL_LENGTH_CONST (allocation of the tool length components to the geometry axes)
- SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type)
- SD42935 $SC_WEAR_TRANSFORM (transformation of wear values)
- SD42960 $SC_TOOL_TEMP_COMP (tool length offsets)

17.10.2 Mirror tool lengths (SD42900 $SC_MIRROR_TOOL_LENGTH)

Activation

Tool length mirroring is activated via the setting data:

SD42900 $SC_MIRROR_TOOL_LENGTH <> 0 (TRUE) (Sign change tool length when mirroring)

Function

The following components are mirrored by inverting the sign:

- Tool lengths: $TC_DP3, $TC_DP4, $TC_DP5
- Tool base dimensions: $TC_DP21, $TC_DP22, $TC_DP23

Mirroring is performed for all tool base dimensions whose associated axes are mirrored. Wear values are **not** mirrored.

Mirror wear values

The following setting data should be set in order to mirror the wear values:

SD42910 $SC_MIRROR_TOOL_WEAR <> 0 (Sign change tool wear when mirroring)

Inverting the sign mirrors the wear values of the tool length components whose associated axes are mirrored.

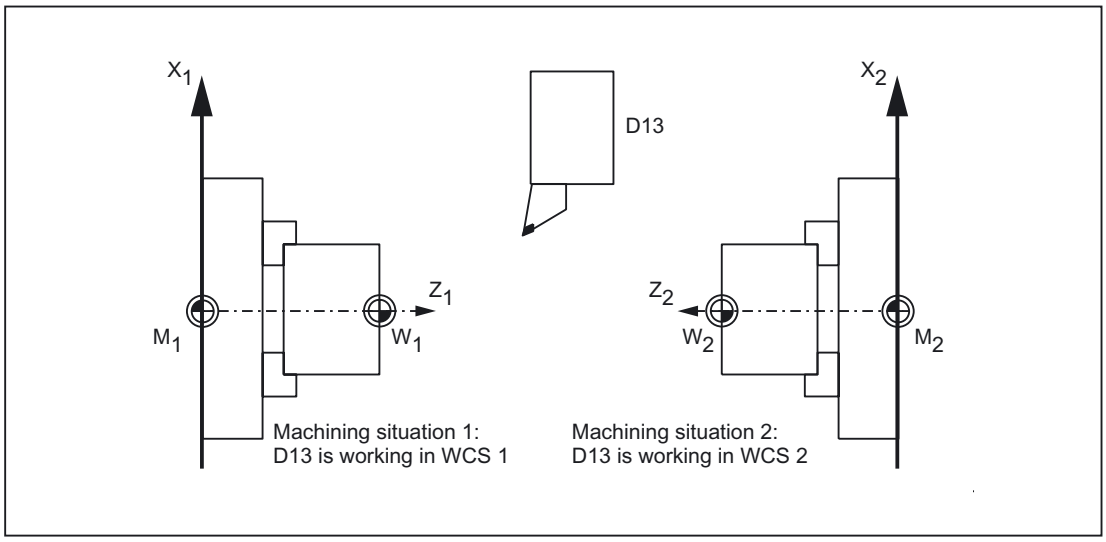


Figure 17-49 Application example: Double-spindle turning machine

17.10.3 Mirror wear lengths (SD42920 $SC_WEAR_SIGN_CUTPOS)

Activation

Wear length mirroring is activated by:

SD42920 $SC_WEAR_SIGN_CUTPOS <> 0 (TRUE) (Sign of wear for tools with cutting edge systems)

Function

Length of cutting edge	Length 1	Length 2
1	---	---
2	---	Inverted
3	Inverted	Inverted
4	Inverted	---
5	---	---
6	---	---
7	---	Inverted
8	Inverted	---
9	---	---

In the case of tool types without a relevant cutting edge position, the wear length is not mirrored.

Note

The mirroring (sign inversion) in one or more components can cancel itself through a simultaneous activation of the functions:

Tool length-mirroring (SD42900 <> 0)

And:

Tool length-mirroring (SD42920 <> 0)

SD42930 $SC_WEAR_SIGN

Setting data not equal to zero:

Inverts the sign of all wear dimensions. This affects both the tool length and other variables such as tool radius, rounding radius, etc.

Entering a positive wear dimension makes the tool "shorter" and "thinner".

Activation of modified setting data

When the setting data described above are modified, the tool components are not reevaluated until the next time a tool edge is selected. If a tool is already active and the data of this tool are to be reevaluated, the tool must be selected again.

Example:

```
N10 $SC_WEAR_SIGN = 0              ; No sign inversion of the wear values
N20 $TC_DP1[1,1] = 120             ; End mill
N30 $TC_DP6[1,1] = 100             ; Tool radius 100 mm
N40 $TC_DP15[1,1] = 1              ; Wear dimension of tool radius 1 mm, resulting
                                     tool radius 101 mm
N100 T1 D1 G41 X150 Y20
....
N150 G40 X300N10
....
N200 $SC_WEAR_SIGN = 1             ; Sign inversion for all wear values; the new
                                     radius of 99 mm is activated on a new
                                     selection (D1). Without D1, the radius would
                                     continue to be 101 mm.
N300 D1 G41 X350 Y-20
N310 ....
```

The same applies in the event that the resulting tool length is modified due to a change in the mirroring status of an axis. The tool must be selected again after the mirror command, in order to activate the modified tool-length components.

17.10.4 Tool length and plane change (SD42940 $SC_TOOL_LENGTH_CONST)

Plane change

The assignment of tool length components (length, wear and tool base dimension) to geometry axes does not change when the machining plane is changed (G17-G19).

Assignment of tools

The assignment of tool length components to geometry axes for turning and grinding tools (tool types 400 to 599) is generated from the value of the following setting data in accordance with the following table:

SD42940 $SC_TOOL_LENGTH_CONST (change of tool length components on change of planes).

Layer	Length 1	Length 2	Length 3
17	Y	X	Z
*)	X	Z	Y
19	Z	Y	X
-17	X	Y	Z
-18	Z	X	Y
-19	Y	Z	X
*) Each value not equal to 0, which is not equal to one of the six listed values, is evaluated as value 18.			

The following table shows the assignment of tool length components to geometry axes for **all other tools** (tool types < 400 or > 599):

Layer	Length 1	Length 2	Length 3
*)	Z	Y	X
18	Y	X	Z
19	X	Z	Y
-17	Z	X	Y
-18	Y	Z	X
-19	X	Y	Z
*) Each value not equal to 0, which is not equal to one of the six listed values, is evaluated as value 17.			

Note

For representation in tables, it is assumed that geometry axes 1 to 3 are named X, Y, Z. The axis order and not the axis identifier determines the assignment between a compensation and an axis.

Three tool length components can be arranged on the 6 different types above.

17.10.5 Tool type (SD42950 $SC_TOOL_LENGTH_TYPE)

Characteristics

Definition of the assignment between tool length components (length, wear and tool base dimension) and geometry axes independent of tool type.

Setting data **not equal to** zero: (the default definition is applied)

A distinction is made between turning and grinding tools (tool types 400 to 599) and other tools (milling tools).

The value range is from 0 to 2. Any other value is interpreted as 0.

The assignment of tool length components is always independent of the actual tool type.

- With value = 1: Always as for milling tools
- With value = 2: Always as for turning tools

Toolholder with orientation capability

Setting data SD42900 - SD42950

Setting data SD42900 - SD42950 have no effect on the components of an active toolholder with orientation capability. The calculation with a toolholder with orientation capability always allows for a tool with its total resulting length (tool length + wear + tool base dimension). The calculation of the resulting total length allows for all modifications caused by the setting data.

Note

When toolholders with orientation capability are used, it is common to define all tools for a non-mirrored basic system, even those, which are only used for mirrored machining. When machining with mirrored axes, the toolholder is then rotated such that the actual position of the tool is described correctly. All tool-length components then automatically act in the correct direction, dispensing with the need for control of individual component evaluation via setting data, depending on the mirroring status of individual axes.

The use of toolholders with orientation capability is also practical if the physical characteristics of the machine type prevents tools, which are permanently installed with different orientations, from being rotated. Tool dimensioning can then be performed uniformly in a basic orientation, where the dimensions relevant for machining are calculated according to the rotations of a virtual toolholder.

17.10.6 Tool lengths in the WCS, allowing for the orientation

Change tool or working plane

The values displayed for the tool correspond to the expansion in the WCS. If a toolholder with an inclined clamping position is to be used, you should make sure that the transformation used supports the toolholder. If this is not the case, incorrect tool dimensions will be displayed. When changing the working plane from G17 to G18 or G19, you should ensure that the transformation can also be used for these working planes. If the transformation is only available for G17 machining, the dimensions continue to be displayed for a tool in the Z direction after the plane change.

When transformation is deactivated, the basic tool is displayed in the x, y or z direction, according to the working plane. Allowance is made for a programmed toolholder. These tool dimensions are not altered when traversing without a transformation.

17.10.7 Tool length offsets in tool direction

Temperature compensation in real time

On 5-axis machines with a moving tool, temperature fluctuations can occur in the machining heads. These can result directly in expansion fluctuations which are transmitted to the tool spindle in the form of linear expansion. A typical case on 5-axis heads, for example, is thermal expansion in the direction of the longitudinal spindle axis.

It is possible to compensate this thermal expansion even when the tool is orientated by assigning the temperature compensation values to the tool rather than to the machine axes. In this way, linear expansion fluctuations can be compensated even when the tool orientation changes.

Using the orientation transformation whose direction is determined by the current tool orientation, it is possible to overlay motions in real time and rotate them simultaneously. At the same time, the compensation values are adjusted continuously in the tool coordinate system.

The temperature compensation only becomes effective if the axis to be compensated is really referenced.

Activation

The temperature compensation in the tool direction is activated by setting the following machine data to a value **not equal to zero**.

MD20390 $MC_TOOL_TEMP_COMP_ON (activation of temperature compensation for tool length)

In addition, bit 2 must be set for each affected channel axis in the machine data:

MD32750 $MA_TEMP_COMP_TYPE [<axis index>] (temperature compensation type)

This can be more than three axes in cases where more than three channel axes in succession can be temporarily assigned to geometry axes as a result of geometry axis replacement of transformation switchover. If this bit is not set for a particular channel axis, the compensation value cannot be applied in the axis. This does not have any effect on other axes. In this case, an alarm is not output.

Applicability

The temperature compensation in the tool direction is only effective with generic 5-axis transformations with:

- Transformation type 24

  Two axes rotate the tool

- Transformation type 56

  One axis rotates the tool, the other axis rotates the workpiece without temperature compensation

In generic 5-axis transformation with:

- Transformation type 40

  The tool orientation is constant with a rotary workpiece, which means that the motion of the rotary axes on the machine does not affect the temperature compensation direction.

Temperature compensation in the tool direction also works in conjunction with orientation transformations (not generic 5-axis transformations) with:

- Transformation type 64 to 69

  Rotating linear axis

Note

Temperature compensation can be activated with all other types of transformation. It is not affected by a change in tool orientation. The axes move as if no orientation transformation with temperature compensation were active.

Limit values

The compensation values are restricted to the maximum values by the machine data:

MD20392 $MC_TOOL_TEMP_COMP_LIMIT[0...2] (maximum temperature compensation for tool length)

The limit value default setting is 1 mm. If a temperature compensation value higher than this limit is specified, it will be limited without an alarm.

SD42960

The three temperature compensation values together form a compensation vector and are contained in setting data:

SD42960 $SC_TOOL_TEMP_COMP[0...2] (temperature compensation with reference to tools)

The setting data is user-defined, e.g. using synchronized actions or from the PLC. The compensation values can, therefore, also be used for other compensation purposes.

In the initial state or when orientation transformation is deactivated, all three compensation values apply in the direction of the three geometry axes (in the typical order X, Y, Z). The assignment of components to geometry axes is independent of the tool type (turning, milling or grinding tools) and the selected machining plane `G17` to `G19`. Changes to the setting data values take effect immediately.

Toolholder with orientation capability

If a toolholder with orientation capability is active, the temperature compensation vector is rotated simultaneously to any change in orientation. This applies independently of any active orientation transformation.

If a toolholder with orientation capability is active in conjunction with a generic 5-axis transformation or a transformation with rotating linear axis, the temperature compensation vector is subjected to both rotations.

Note

While transformations with rotating linear axes take changes in the tool vector (length) into account, they **ignore** its change in orientation, which can be effected by a toolholder with orientation capability.

Temperature compensation values immediately follow any applied change in orientation. This applies in particular when an orientation transformation is activated or deactivated.

The same is true when the assignment between geometry axes and channel axes is changed. The temperature compensation value for an axis is reduced to zero (interpolatively), for example, when it ceases to be a geometry axis after a transformation change. Conversely, any temperature compensation value for an axis which changes over to geometry axis status is applied immediately.

Examples

Temperature compensation in tool direction

Example of a 5-axis machine with rotating tool on which the tool can be rotated around the C and B axes.

In its initial state, the tool is parallel to the Z axis. If the B axis is rotated through 90 degrees, the tool points in the X direction.

Therefore, a temperature compensation value in the following setting data is also effective in the direction of the machine X axis if transformation is active:

SD42960 $SC_TOOL_TEMP_COMP[2] (temperature compensation with reference to tools)

If the transformation is deactivated with the tool in this direction, the tool orientation is, by definition, parallel again to the Z axis and thus different to its actual orientation. The temperature offset in the X axis direction is therefore reduced to zero and reapplied simultaneously in the Z direction.

Example of a 5-axis machine with rotating tool (transformation type 24). The relevant machine data is listed below:

- The first rotary axis rotates around Z, C-axis
- The second rotary axis rotates around Y, B-axis

The essential machine data is shown in the table below:

Machine data	Value	Remark
MD20390 $MC_TOOL_TEMP_COMP_ON	= TRUE	Temperature compensation active
MD32750 $MA_TEMP_COMP_TYPE[AX1]	= 4	Compensation in tool direction
MD32750 $MA_TEMP_COMP_TYPE[AX2]	= 4	Compensation in tool direction
MD32750 $MA_TEMP_COMP_TYPE[AX3]	= 4	Compensation in tool direction
		Assignment of transformation type 24
MD24100 $MC_TRAFO_TYPE_1	= 24	Transformer type 24 in first channel
MD24110 $MC_TRAFO_AXES_IN_1[0]	= 1	First axis of the transformation
MD24110 $MC_TRAFO_AXES_IN_1[1]	= 2	Second axis of the transformation
MD24110 $MC_TRAFO_AXES_IN_1[2]	= 3	Third axis of the transformation
MD24110 $MC_TRAFO_AXES_IN_1[3]	= 5	Fifth axis of the transformation
MD24110 $MC_TRAFO_AXES_IN_1[4]	= 4	Fourth axis of the transformation
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]	= 1	Geometry axis for channel axis 1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]	= 2	Geometry axis for channel axis 2
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]	= 3	Geometry axis for channel axis 3
MD24570 $MC_TRAFO5_AXIS1_1[0]	= 0.0	
MD24570 $MC_TRAFO5_AXIS1_1[1]	= 0.0	Direction
MD24570 $MC_TRAFO5_AXIS1_1[2]	= 1.0	First rotary axis is parallel to Z
MD24572 $MC_TRAFO5_AXIS1_2[0]	= 0.0	Direction
MD24572 $MC_TRAFO5_AXIS1_2[1]	= 1.0	Second rotary axis is parallel to Y
MD24572 $MC_TRAFO5_AXIS1_2[2]	= 0.0	
MD25574 $MC_TRAFO5_BASE_ORIENT_1[0]	= 0.0	
MD25574 $MC_TRAFO5_BASE_ORIENT_1[1]	= 0.0	Basic tool orientation
MD25574 $MC_TRAFO5_BASE_ORIENT_1[2]	= 1.0	In Z direction

Temperature compensation values in the NC program

The compensation values assigned to axes X and Z are not zero and are applied for temperature compensation with respect to tool length. The machine axis positions reached in each case are specified as comments in the program lines.

```
Program code                                    Comment
$SC_TOOL_TEMP_COMP[0] = -0.3                    ; Compensation value in X
$SC_TOOL_TEMP_COMP[1] = 0.0                     ;
$SC_TOOL_TEMP_COMP[2] = -1.0                    ; Compensation value in Z
                                                ; Position setpoints of the machine
                                                  axes
N10 G74 X0 Y0 Z0 A0 B0                          ; X Y Z
N20 X20 Y20 Z20 F10000                          ; 20.30 20.00 21.00
N30 TRAORI()                                    ; 20.30 20.00 21.00
N40 X10 Y10 Z10 B90                             ; 11.00 10.00 9.70
N50 TRAFOOF                                     ; 10.30 10.00 11.00
N60 X0 Y0 Z0 B0 C0                              ; 0.30 0.00 1.00
N70 M30
```

With the exception of block N40, temperature compensation always acts in the original directions, as the tool is pointing in the basic orientation direction. This applies particularly in block N50. The tool is actually still pointing in the direction of the X axis because the B axis is still at 90 degrees. However, because the transformation is already deactivated, the applied orientation is parallel to the Z axis again.

Machine data	Value	Remark
MD20390 $MC_TOOL_TEMP_COMP_ON	= TRUE	Temperature compensation active
MD32750 $MA_TEMP_COMP_TYPE[AX1]	= 4	Compensation in tool direction
MD32750 $MA_TEMP_COMP_TYPE[AX2]	= 4	Compensation in tool direction
MD32750 $MA_TEMP_COMP_TYPE[AX3]	= 4	Compensation in tool direction

Additional references

For more details on "Temperature compensation" see:

References:
Function Manual Extended Functions; Compensations (K3)

For information on "Generic 5-axis transformations" see:

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

17.11 Sum offsets and setup offsets

17.11.1 General

Sum offsets

Sum offsets can be treated as **programmable process compensations** during machining and are composed of all the error sizes (including the wear), which cause the workpiece to deviate from the specified dimensions.

Sum offsets are a generalized type of wear. They are part of the cutting edge data. The parameters of the sum offset refer to the geometrical data of a cutting edge.

The compensation data of a sum offset are addressed by a **DL number** (**DL:** location-dependent; compensations with reference to the location of use).

In contrast, the wear values of a D number describe the physical wear of the cutting edge, i.e., in special situations, the sum offset can match the wear of the cutting edge.

Sum offsets are intended for general use, i.e., with active or inactive tool management or with the flat D number function.

Machine data are used to classify the sum offsets into:

- Sum offset fine
- Sum offset coarse (setup offset)

Setup offset

The setup offset is the compensation to be entered by the setup engineer before machining. These values are stored separately in the NCK. The operator subsequently only has access to the "sum offset fine" via HMI.

The "sum offset fine" and "sum offset coarse" are added internally in the NCK. This value is referred to below as the sum offset.

Note

The function is enabled via the machine data setting:

MD18080 $MN_MM_TOOL_MANAGEMENT_MASK, Bit 8=1 (Gradual memory reservation for tool management).

If kinematic transformations (e.g., 5axis transformations) are active, the tool length is calculated first after allowing for the various wear components. ´The total tool length is then used in the transformation. Unlike the case of a toolholder with orientation capability, the wear values are thus always included in the transformation irrespective of the G code of group 56.

17.11.2 Description of function

Sum offsets

Several sum offsets (DL numbers) can be defined per D number. This allows you to determine, for example, **workpiecelocationdependent** compensation values and assign them to a cutting edge. Sum offsets have the same effect as wear, i.e., they are added to the compensation values of the D number. The data are permanently assigned to a D number.

Attitudes

You can define the following settings in machine data:

- Activate sum offset
- Define maximum quantity of DL data sets to be created in NCK memory
- Define maximum quantity of DL numbers to be assigned to a D number
- Define whether the sum offsets (fine/coarse) are to be saved during data backup
- Define the sum offset to be activated, if:
  - A new cutting edge compensation is activated
  - An operator panel front RESET is performed
  - An operator panel front START is performed
  - The end of the program has been reached

The name is oriented to the logic of the corresponding machine data for tools and cutting edges.

The "setup offset" and "sum offset fine" can be read and written via system variables and corresponding OPI services.

Note

When tool management is active, a machine data can be used to define whether the sum offset of a tool activated during a programmed tool change remains unchanged or is set to zero.

Summary of compensation parameters $TC_DPx

The following general system variables were previously defined for describing a cutting edge:

$TC_DP1	Tool type
$TC_DP2	Length of cutting edge

Parameters for geometry and wear

Tool geometry compensations are assigned to system variables $TC_DP3 to $TC_DP11. System variables $TC_DP12 to $TC_DP20 allow you to name a wear for each of these parameters.

Geometry	Wear	Length compensations
$TC_DP3	$TC_DP12	Length 1
$TC_DP4	$TC_DP13	Length 2
$TC_DP5	$TC_DP14	Length 3
Geometry	Wear	Radius compensation
$TC_DP6	$TC_DP15	Radius
$TC_DP7	$TC_DP16	Corner radius (tool type 700; slotting saw)
Geometry	Wear	Further compensations
$TC_DP8	$TC_DP17	Length 4 (tool type 700; slotting saw)
$TC_DP9	$TC_DP18	Length 5
$TC_DP10	$TC_DP19	Angle 1 (angle between face of tool and torus surface)
$TC_DP11	$TC_DP20	Angle 2 (angle between tool longitudinal axis and upper end of torus surface)

Tool base dimension/adapter dimension

$TC_DP21	Adapter length 1
$TC_DP22	Adapter length 2
$TC_DP23	Adapter length 3

Technology

System variable	Clearance angle
$TC_DP24	• The clearance angle is stored here for ManualTurn; tool type 5xx. Same significance as in standard cycles for turning tools. • The tip angle of the drill is stored here for ShopMill; tool type 2xx. • Used in standard cycles for turning tools; tool type 5xx. This is the angle at the secondary cutting edge for these tools.
$TC_DP25	• The value for the cutting rate is stored here for ManualTurn. • A bitcoded value for various states of tool types 1xx and 2xx is stored here for ShopMill.

Parameters of the sum and setup offsets ($TC_SCPxy, $TC_ECPxy)

The numbering of the parameters is oriented to the numbering of system variables $TC_DP3 to $TC_DP11.

The effect of the parameters is similar to the wear (additive to the tool geometry). Up to six sum/setup parameters can be defined per cutting edge parameter.

Tool geometry parameter, to which the compensation is added.	Sum/setup parameters, length compensations	Tool wear parameters
$TC_DP3	Length 1 $TC_SCP**1**3, $TC_SCP**2**3,$TC_SCP**3**3, $TC_SCP**4**3,$TC_SCP**5**3,$TC_SCP**6**3 $TC_ECP**1**3, $TC_ECP**2**3,$TC_ECP**3**3, $TC_ECP**4**3,$TC_ECP**5**3,$TC_ECP**6**3 The numbers in bold, 1, 2, ... 6, designate the parameters of a maximum of six (location-dependent or similar) compensations that can be programmed with DL =1 to 6 for the parameter specified in column one.	$TC_DP12
$TC_DP4	Length 2 $TC_SCP**1**4, $TC_SCP**2**4,$TC_SCP**3**4, $TC_SCP**4**4,$TC_SCP**5**4,$TC_SCP**6**4 $TC_ECP**1**4, $TC_ECP**2**4,$TC_ECP**3**4, $TC_ECP**4**4,$TC_ECP**5**4,$TC_ECP**6**4	$TC_DP13
$TC_DP5	Length 3 etc.	$TC_DP14
	Radius compensation	
$TC_DP6	Radius	$TC_DP15
$TC_DP7	Corner radius	$TC_DP16
	Further compensations	
$TC_DP8	Length 4	$TC_DP17
$TC_DP9	Length 5	$TC_DP18
$TC_DP10	Angle 1, etc.	$TC_DP19
$TC_DP11	Angle 2 $TC_SCP**2**1, $TC_SCP**3**1,$TC_SCP**4**1, $TC_SCP**5**1,$TC_SCP**6**1,$TC_SCP**7**1 $TC_ECP**2**1, $TC_ECP**3**1,$TC_ECP**4**1, $TC_ECP**5**1,$TC_ECP**6**1,$TC_ECP**7**1 The numbers in bold, 2, 3, ... 7, designate the parameters of a maximum of six (location-dependent or similar) compensations that can be programmed with DL =1 to 6 for the parameter specified in column one.	$TC_DP20

Supplementary conditions

The maximum number of DL data sets of a cutting edge and the total number of sum offsets in the NCK are defined by machine data. The default value is zero, i.e., no sum offsets can be programmed.

Activate the "monitoring function" to monitor a tool for wear or for "sum offset".

The additional sum/setup data sets use additional buffered memory. 8 bytes are required per parameter.

A sum-offset data set requires: 8 bytes * 9 parameters = 72 bytes

A setup data set requires an equal amount of memory. A certain number of bytes is also required for internal administration data.

17.11.3 Activation

Function

The function must be activated via the machine data:

MD18108 $MN_MM_NUM_SUMCORR (sum offsets in TO area).

System variables $TC_ECPx and $TC_SCPx and setup and sum offsets ("fine") defined via the OPI interface can be activated in the part program.

This is done by programming the language command `DL="number"`.

When a new D number is activated, either a new DL number is programmed, or the DL number defined via the following machine data becomes active:

MD20272 $MC_SUMCORR_DEFAULT (basic setting of the additive offset without a program)

DL programming

The sum offset is always programmed relative to the active D number with the command:

`DL =` "n"

The sum offset "n" is added to the wear of the active D number.

Note

If you use "setup offset" **and** "sum offset fine", both compensations are combined and added to the tool wear.

The sum offset is deselected with the command:

```
DL = 0
```

Note

DL0 is not allowed. If compensation is deselected (`D0` and `T0`), the sum offset also becomes ineffective.

Programming a sum offset that does not exist triggers an alarm, similar to programming a D compensation that does not exist.

Thus, only the defined wear remains part of the compensation (defined in system variables $TC_DP12 to $TC_DP20).

Programming a sum offset when a D compensation is active (also applies to deselection) has the same effect on the path as programming a D command. An active radius compensation will, therefore, lose its reference to adjacent blocks.

Configuration

MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4=0: (Properties of sum offset in the TO area) default setting:

Only **one** set of sum offsets exists per DL number.

We refer in general to the sum offset.

This describes the data represented by $TC_SCPx.

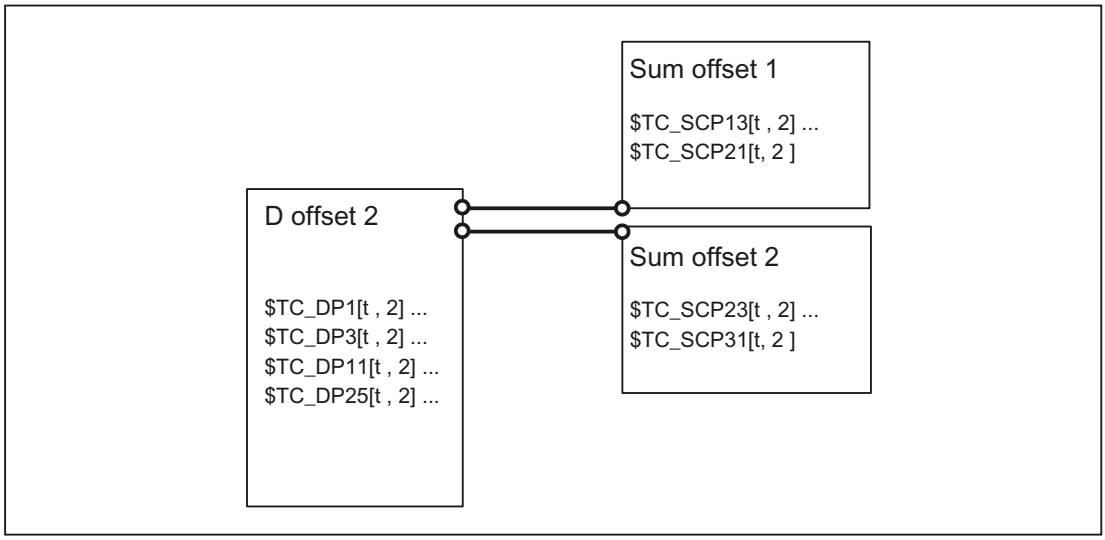


Figure 17-50 MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4 = 0

Tool T = t is active. With the data in the figure, the following is programmed:

```
D2                  ;  Cutting edge offsets, i.e., $TC_DP3 to $TC_DP11 + wear
                       ($TC_DP12 to $TC_DP20) + adapter dimension
...
DL=1                ;  Sum offset 1 is added to the previous D2 compensations, i.e.,
                       $TC_SCP13 to $TC_SCP21.
...
DL=2                ;  Sum offset 2 is added to the D2 compensation instead of sum
                       offset 1, i.e., $TC_SCP23 to $TC_SCP31.
...
DL=0                ;  Deselection of sum offset;
                       only the data of D2 remain active.
```

MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4=1: Setup offsets are available

The sum offset is now composed of the "sum offset fine" (represented by $TC_SCPx) and the setup offset (represented by $TC_ECPx). Two data sets therefore exist for one DL number. The sum offset is calculated by adding the corresponding components ($TC_ECPx + $TC_SCPx).

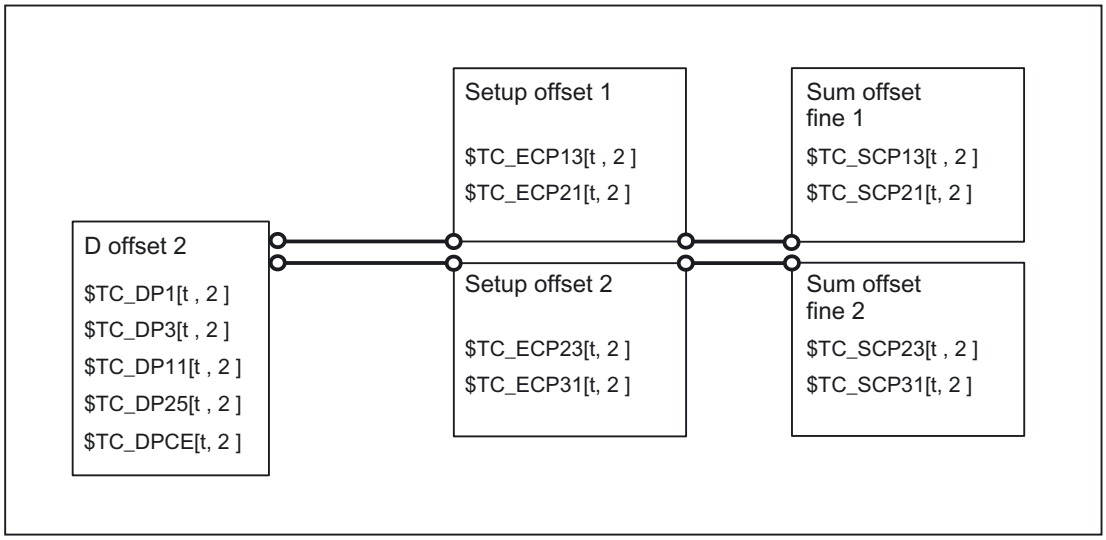


Figure 17-51 MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4 = 1 "setup offsets" + "sum offsets fine"

Tool T = t is active. With the data in the figure, the following is programmed:

```
D2                ;  Cutting edge compensations, i.e., $TC_DP3 to $TC_DP11 + wear
                     ($TC_DP12 to $TC_DP20) + adapter dimension
...
DL=1              ;  Sum offset 1 is added to the previous D2 compensations, i.e.,
                     $TC_ECP13 + $TC_SCP13 to $TC_ECP21 + $TC_SCP21.
...
DL=2              ;  Sum offset 2 is added to the D2 compensation instead of sum
                     offset 1; i.e., $TC_ECP23 + $TC_SCP23,...$TC_ECP31 + $TC_SCP31
...
DL=0              ;  Deselection of sum offset. Only the data of D2 remain active.
```

Reading/writing in the part program

The individual sets of sum offset parameters are differentiated according to the number ranges of system variable $TC_SCP.

The significance of the individual variables is similar to geometry variables $TC_DP3 to $TC_DP11. Only length 1, length 2 and length 3 are enabled for the basic functionality (variables $TC_SCP13 to $TC_SCP15 for the first sum offset of the cutting edge).

```
R5 = $TC_SCP13[ t, d ]          ; Sets the value of the R parameter to the value
                                  of the first component of sum offset 1 for
                                  cutting edge (d)
                                   on tool (t).
R6 = $TC_SCP21[ t, d ]          ; Sets the value of the R parameter to the value
                                  of the last component of sum offset 1 for
                                  cutting edge (d) on tool (t).
R50 = $TC_SCP23[ t, d ]         ; Sets the value of the R parameter to the value
                                  of the first component of sum offset 2 for
                                  cutting edge (d) on tool (t).
$TC_SCP43[ t, d ] = 1.234       ; Sets the value of the first component of sum
                                  offset 4 for cutting edge (d) on tool (t) to the
                                  value 1.234.
```

The above statements also apply to the setup offsets (if the NCK is configured with this option), i.e.,

```
R5 = $TC_ECP13[ t, d ]          ; Sets the value of the R parameter to the value
                                  of the first component of setup offset 1 for
                                  cutting edge (d) on tool (t).
R6 = $TC_ECP21[ t, d ]          ; Sets the value of the R parameter to the value
                                  of the last component of setup offset 1 for
                                  cutting edge (d) on tool (t).

Etc.
```

When working with setup offsets, "sum offsets fine" are written with the $TC_SCPx system variables.

Creating a new sum offset

If the compensation data set (x) does not yet exist, it is created on the first write operation to one of its parameters (y).

```
$TC_SCPxy[ t, d ] = r.r          ; Parameter y of sum offset x is assigned the
                                   value "r.r.". The other parameters of x have a
                                   value of zero.
```

When working with setup offsets, "sum offsets fine" are written with the $TC_SCPx system variables.

Note

When working with setup offsets, the data set for the setup offset is created when a data set is created for "sum offset fine", if a data set did not already exist for [t, d].

Creating a new setup offset

If the compensation data set (x) does not yet exist, it is created on the first write operation to one of its parameters (y).

```
$TC_ECPxy[ t, d ] = r.r          ; The value "r.r" is assigned to the parameter y
                                   of setup offset x. The other parameters of x
                                   have the value zero.
```

Note

When working with setup offsets, the data set for the "sum offset fine" is created when a data set is created for setup offsets, if a data set did not already exist for [t, d].

DELDL - Delete sum offset

Sum offsets are generally only relevant when machining with a cutting edge at a certain time at a certain location of the workpiece. You can use the NC language command DELDLto delete sum offsets from cutting edges (in order to release memory).

```
status = DELDL( t, d )          ; Deletes all sum offsets for cutting edge d on tool
                                  t.
                                ; t, d are optional parameters.
```

If d is not specified, all sum offsets of all cutting edges of tool t are deleted.

If d and t are not specified, all sum offsets for the cutting edges on all tools of the TO unit are deleted (for the channel, in which the command is programmed).

When working with setup offsets, the DELDL command deletes both the setup offset and the "sum offsets fine" of the specified cutting edge(s).

Note

The memory used for the data sets is released after deletion.

The deleted sum offsets can subsequently no longer be activated or programmed.

Sum offsets and setup offsets on active tools cannot be deleted (similar to the deletion of D compensations or tool data).

The "status" return value indicates the result of the deletion command:

0:	Deletion was successful
-1:	Deletion was not (one cutting edge) or not completely (several cutting edges) successful

Data backup

The data are saved during a general tool-data backup (as a component of the D number data sets).

It is advisable to save the sum offsets, in order to allow the current status to be restored in the event of an acute problem. Machine data settings can be made to exclude sum offsets from a data backup (settings can be made separately for "setup offsets" and "sum offsets fine").

Note

Sum offsets behave in the same way as D compensations with reference to block search and REPOS. The behavior on Reset and PowerOn can be defined by machine data.

If the setting of the following machine data indicates that the last active tool compensation number (D) is to be activated after PowerOn, the last active DL number is then no longer active:

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

17.11.4 Examples

Example 1

That no compensation and no sum offset will come into effect must be defined during tool change via the machine data:

- MD20270 $MC_CUTTING_EDGE_DEFAULT=0 (Basic setting of tool cutting edge without programming)
- MD20272 $MC_SUMCORR_DEFAULT=0 (default setting sum offset without program).

```
T5 M06              ; Tool number 5 is loaded - no compensation active.
D1 DL=3             ; Compensation D1 + sum offset 3 of D1 are activated.
X10
DL=2                ; Compensation D1 + sum offset 2 are activated.
X20
DL=0                ; Sum offset deselection, only compensation D1 is now active.
D2                  ; Compensation D2 is activated - the sum offset is not included
                      in the compensation.
X1
DL=1                ; Compensation D2 + sum offset 1 are activated.
X2
D0                  ; Compensation deselection
X3
DL=2                ; No effect - DL2 of D0 is zero (same as programming T0 D2).
```

Example 2

During tool change it has to be defined that offset D2 and sum offset DL=1 are activated via the machine data:

MD20270 $MC_CUTTING_EDGE_DEFAULT=2 (Basic setting of tool cutting edge without programming)

MD20272 $MC_SUMCORR_DEFAULT=1 (default setting sum offset without program)

```
T5 M06              ;  Tool number 5 is loaded - D2 + DL=1 are active (= values of
                       machine data)
D1 DL=3             ;  Compensation D1 + sum offset 3 of D1 are activated.
X10
DL=2                ;  Compensation D1 + sum offset 2 are activated.
X20
DL=0                ;  Sum offset deselection, only compensation D1 is now active.
D2                  ;  Compensation D2 is activated - sum offset DL=1 is activated.
X1
DL=2                ;  Compensation D2 + sum offset 2 are activated.
D1                  ;  Compensation D1 + sum offset 1 are activated.
```

17.11.5 Upgrades for Tool Length Determination

17.11.5.1 Taking the compensation values into account location-specifically and workpiece-specifically

Composition of the effective tool length

For a tool compensation without active kinematic transformation, the effective tool length consists of up to 8 vectors:

- Tool length (geometry) ($TC_DP3 - $TC_DP5)
- Wear ($TC_DP12 - $TC_DP14)
- Tool base dimension (see note) ($TC_DP21 - $TC_DP23)
- Adapter dimension (see note) ($TC_ADPT1 - $TC_ADPT3)
- Total offsets fine ($TC_SCPx3 - $TC_SCPx5)
- Sum offsets coarse or setup offsets ($TC_ECPx3 - $TC_ECPx5)
- Offset vector l_1 of toolholder with orientation capability ($TC_CARR1 - $TC_CARR3)
- Offset vector l_2 of toolholder with orientation capability ($TC_CARR4 - $TC_CARR6)
- Offset vector l_3 of toolholder with orientation capability ($TC_CARR15 - $TC_CARR17)

Note

The tool base dimension and adapter dimension can only be applied as alternatives.

Type of action of the individual vectors

The type of action of the individual vectors or groups of vectors depends on the following further quantities:

Influencing quantity	Operating principle
G codes	Active machining plane
Tool type	Milling tool or turning/grinding tools
Machine data	Tool management active/not active, toolholder with orientation capability available/not available
Setting data	Behavior of tool length components when mirroring or when changing the plane
Toolholder with orientation capability	Set values of toolholder with orientation capability
Adapter transformations	Transformed tool compensation values

Distribution over the geometry-axis components

How the three vector components of partial totals of the vectors involved are distributed over the three geometry-axis components is determined by the following quantities:

Influencing quantity	Dependencies
Active processing level: G17 X/Y direction G18 Z/X direction G19 Y/Z direction	Infeed plane: Z Y X
Tool type: Milling tools, drilling tools, grinding tools, turning tools	See Section "Tool parameter 1: Tool type (Page 1598)", Table "Minimum number of required tool parameters"
SD42900 $SC_MIRROR_TOOL_LENGTH SD42910 $SC_MIRROR_TOOL_WEAR SD42920 $SC_WEAR_SIGN_CUTPOS SD42930 $SC_WEAR_SIGN SD42940 $SC_TOOL_LENGTH_CONST SD42950 $SC_TOOL_LENGTH_TYPE	See Section "Special handling of tool compensations (Page 1712)" and Section "Setting data (Page 1788)".
Adapter transformations	**References:** Function Manual Tool Management

The resulting tool orientation always remains parallel to one of the three axis directions X, Y or Z and exclusively depends on the active machining plane G17-G19, since it has not yet been possible to assign the tool an orientation.

Stepless variation of the tool orientation

The toolholder with orientation capability also enables the tool orientation to be varied steplessly, in addition to providing further offsets or linear expansion fluctuations with the aid of offset vectors I_1 - I_3 .

For further explanations, see Section "Toolholder with orientation capability (Page 1657)".

Minor operator compensations

Minor compensations, however, must also be modified during the normal production mode.

The reasons for this are, for example:

- Tool wear
- Clamping errors
- Temperature sensitivity of the machine:

These compensations are defined as follows:

Definition	Wear components
Wear	$TC_DP12 - $TC_DP14,
Total offsets fine	$TC_SCPx3 - $TC_SCPx5,
Sum offsets coarse or setup offsets	$TC_ECPx3 - $TC_ECPx5

In particular, compensations, which affect the tool length calculation, should be entered in the coordinates used for measurement.

These workpiece-specific compensations can be achieved more simply using the G-code group 56 with the three values `TOWSTD`, `TOWMCS` and `TOWWCS` and the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformation of tool components)

SD42935

Which of the wear components:

- Wear ($TC_DP12 - $TC_DP14)
- Setup offsets or sum offsets coarse ($TC_ECPx3 - $TC_ECPx5)
- Sum offsets fine ($TC_SCPx3 - $TC_SCPx5)

are to be transformed in the transformations:

- Adapter transformation
- Toolholder with orientation capability

are to be or not to be transformed, can be defined via the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformation of wear values)

With the setting data in its initial state, all wear values are transformed.

The setting data is considered in the following functions:

- Wear values in the machine coordinate system

  Part program instruction: TOWMCS
- Wear values in the workpiece coordinate system

  Part program instruction: TOWWCS

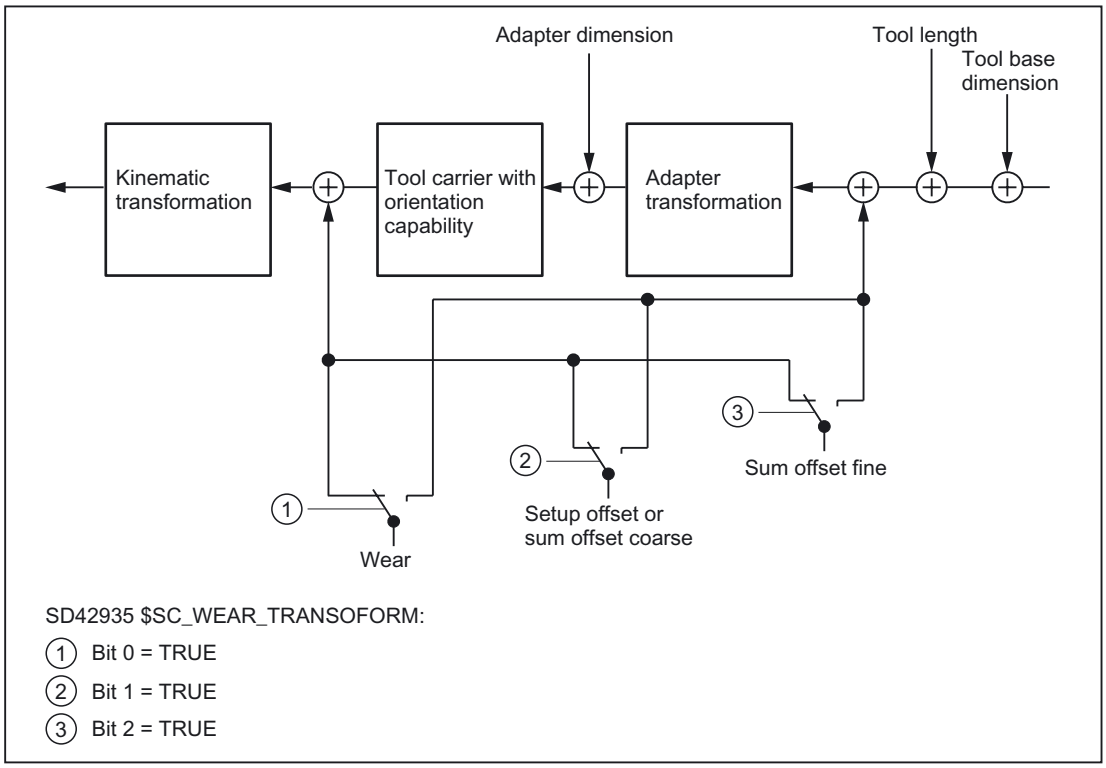


Figure 17-52 Transformation of wear data dependent on SD42935

Programming

G-code group 56 can be used to define the following values:

Syntax	Corrections
TOWSTD	Initial setting value for offsets in tool length
TOWMCS	Wear values in the machine coordinate system (MCS)
TOWWCS	Wear values in the workpiece coordinate system (WCS)
TOWBCS	Wear values in the basic coordinate system (BCS)
TOWTCS	Wear values in the TCS (Tool Coordinate System) at the toolholder (tool carrier reference point T)
TOWKCS	Wear values in tool coordinate system for kinematic transformation (KCS) of tool head

Coordinate systems for offsets in tool length

G codes TOWMCS, TOWWCS, TOWBCS, TOWTCS and TOWKCS can be used, e.g. to measure the wear tool length component in five different coordinate systems.

1. Machine coordinate system	MCS
1. Basic coordinate system	BCS
1. Workpiece coordinate system	WCS
1. Tool coordinate system of kinematic transformation	KCS
1. Tool coordinate system	TCS

The calculated tool length or a tool length component can be represented and read out in one of these coordinate systems using the GETTCOR function (predefined subprogram).

For further explanations, see Section "Read tool lengths, tool length components (Page 1754)".

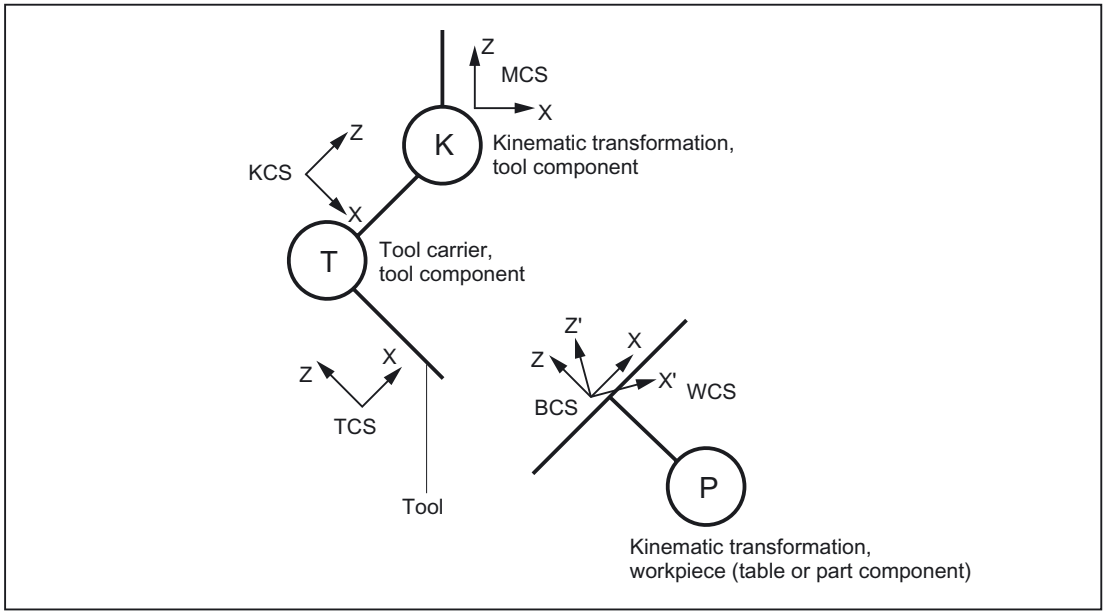


Figure 17-53 Coordinate system for the evaluation of tool lengths

17.11.5.2 Functionality of the individual wear values

TOWSTD

Initial setting (default behavior):

- The wear values are added to the other tool length components.

  The resulting total tool length is then used in further calculations.

In the case of an active toolholder with orientation capability:

- The wear values are subjected to the appropriate rotation.

TOWMCS

Wear data in the MCS (machine coordinate system):

In the case of an active rotation by means of a toolholder with orientation capability:

- The toolholder only rotates the vector of the resultant tool length. Wear is ignored.

  Then the tool length vector rotated in this way and the wear are added. The wear is not subjected to the rotation.

If **no** toolholder with orientation capability is active or this does not result in a rotation, `TOWMCS` and `TOWSTD` are identical.

Linear transformation

The tool length can be uniquely defined in the MCS only if the MCS is generated by linear transformation from the BCS.

This would be the case when:

- No kinematic transformation is active
- Or orientation transformations (3-axis, 4-axis and 5-axis transformations) are active

TOWWCS

Wear values in WCS (workpiece coordinate system):

- If a toolholder with orientation capability is active, the tool vector is calculated as for `TOWMCS`, without taking the wear into account.
- The wear data are interpreted in the workpiece coordinate system.

The wear vector in the workpiece coordinate system is converted to the machine coordinate system and added to the tool vector.

TOWBCS

Wear values in BCS (basic coordinate system):

- If a toolholder with orientation capability is active, the tool vector is calculated as for TOWMCS, without taking the wear into account.
- The wear data are interpreted in the workpiece coordinate system.

The wear vector in the basic coordinate system is converted to the workpiece coordinate system and added to the tool vector.

Non-linear transformation

If a non-linear transformation is active, e.g. with TRANSMIT, and the MCS is specified as the desired coordinate system, the BCS is automatically used instead of the MCS.

Toolholder with orientation capability

A table component of the toolholder with orientation capability, if available, is not applied directly to the coordinate systems, unlike a table (or part) component of the kinematic transformation. A rotation described by such a component is represented in a basic frame or system frame and is thus included in the transition from WCS to BCS.

Kinematic transformation

The table (or part) component of the kinematic transformation is described by the transition from BCS to MCS.

TOWTCS

Wear values in TCS (tool coordinate system):

- If a toolholder with orientation capability is active, the tool vector is calculated as for TOWMCS, without taking the wear into account.
- The wear data are interpreted in the tool coordinate system.

The wear vector in the TCS (Tool Coordinate System) is converted to the machine coordinate system by way of the tool coordinate system of the kinematic transformation (KCS) and added to the tool vector.

TOWKCS

The wear value specifications for the kinematic transformation are interpreted in the associated TCS (Tool Coordinate System).

The wear vector is converted to the machine coordinate system by way of the tool coordinate system of the kinematic transformation and added to the tool vector.

G code change when a tool is active

Changing the G code in the group TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, and TOWKCS does not affect an already active tool, and does not become effective until the next tool is selected.

A new G code of this group will also come into effect if it is programmed in the same block, in which a tool is selected.

Evaluation of individual wear components

Evaluation of individual wear components (assignment to geometry axes, sign evaluation) is influenced by:

- The active plane
- The adapter transformation
- The five setting data shown in the table below

Setting data	Wear components		
SD42910 $SC_MIRROW_TOOL_WEAR	TOWSTD	TOWMCS	TOWWCS
SD42920 $SC_WEAR_SIGN_CUTPOS	X	X	—
SD42930 $SC_WEAR_SIGN	X	—	—
SD42940 $SC_TOOL_LENGTH_CONST	X	X	X
SD42950 $SC_TOOL_LENGTH_TYPE	X	X	X

Note

Wear components which are subjected to an active rotation by an adapter transformation or a toolholder with orientation capability are referred to as non-transformed wear components.

Special features

If TOWMCS or TOWWCS is active, the following setting data does not affect the non-transformed wear components:

SD42920 $SC_WEAR_SIGN_CUTPOS (Sign of wear for tools with cutting edge systems)

The following setting data also does not affect the non-transformed wear components in case of TOWWCS:

SD42910 $SC_MIRROR_TOOL_WEAR (Sign change tool wear when mirroring)

In this case, a possibly active mirroring is already contained in the frame, which is referred to for evaluating the wear components.

On a plane change, the assignment between the non-transformed wear components and the geometry axes is retained, i.e. these are not interchanged as with other length components. The assignment of components depends on the active plane for tool selection.

Example

Let's assume a milling tool is used where only the wear value $TC_DP12 assigned to length L1 is not equal to zero.

If G17 is active, this length is effective in the direction of the Z axis.

This measure always acts in the Z-direction also upon a plane change after the tool selection, when TOWMCS or TOWWCS are active and the bit 1 is set in the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)

If, for example, G18 is active on tool selection, the component is always effective in the Y direction instead.

17.12 Working with tool environments

17.12.1 General

Functions

The current states of tool data can be processed using the functions below, which are generally available:

- Save
- Deletion
- Read
- Modify

A further function can be used to determine information about the assignment of the tool lengths of the active tool to the abscissa, ordinate and applicate.

17.12.2 Saving with TOOLENV

Scope of a tool environment

The TOOLENV memory function is used to save any current states needed for the evaluation of tool data stored in the memory.

The individual data are as follows:

- The active G code of group 6 (G17,G18,G19)
- The active G code of group 56 (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS)
- The active transverse axis
- Machine data:

  MD18112 $MN_MM_KIND_OF_SUMCORR (Properties of sum offsets in the TO area)
- Machine data:

  MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).
- Setting data:

  SD42900 $SC_MIRROR_TOOL_LENGTH (Sign change tool length when mirroring)
- Setting data:

  SD42910 $SC_MIRROR_TOOL_WEAR (Sign change tool wear when mirroring)
- Setting data:

  SD42920 $SC_WEAR_SIGN_CUTPOS (Sign of wear for tools with cutting edge systems)

- Setting data:

  SD42930 $SC_WEAR_SIGN (sign of wear)

- Setting data:

  SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)

- Setting data:

  SD42940 $SC_LENGTH_CONST (change of tool components on change of planes)

- Setting data:

  SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type)

- The orientation component of the current complete frame (rotation and mirroring, no work offsets or scales)
- The orientation component and the resulting length of the active toolholder with orientation capability
- The orientation component and the resulting length of an active transformation
- In addition to the data describing the environment of the tool, the T number, D number and DL number of the active tool are also stored, so that the tool can be accessed later in the same environment as the TOOLENV call, without having to name the tool again.

Not in the tool environment

The value of the machine data determines whether the adapter length or the tool base dimension is included in the tool length calculation:

MD18104 $MN_MM_NUM_TOOL_ADAPTER (tool adapter in TO area).

Since a change to this machine data only takes effect after Power On, it is not saved in the tool environment.

Note

Resulting length of **toolholders with orientation capability** and **transformations**:

Both toolholders with orientation capability and transformations can use system variables or machine data, which act as additional tool length components, and which can be subjected partially or completely to the rotations performed. The resulting additional tool length components must also be stored when TOOLENV is called, because they represent part of the environment, in which the tool is used.

Adapter transformation:

The adapter transformation is a property of the tool adapter and thus of the complete tool. It is, therefore, not part of a tool environment, which can be applied to another tool.

By saving the complete data necessary to determine the overall tool length, it is possible to calculate the effective length of the tool at a later point in time, even if the tool is no longer active or if the conditions of the environment (e.g., G codes or setting data) have changed. Similarly, the effective length of different tool can be calculated assuming that it would be used under the same conditions as the tool, for which the status was saved.

TOOLENV function

Saving a tool environment

The TOOLENV function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Syntax:

Status = TOOLENV(_NAME)

Value/parameter:

Status INT

0: Function OK

-1: No memory reserved for tool environments:

MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).

i.e. the "tool environments" functionality is not available.

-2: No more free memory locations for tool environments available.

-3: Null string illegal as name of a tool environment.

-4: No parameter (name) specified.

_NAME STRING

Name, under which the current data set is stored.

If a data set of the same name already exists, it is overwritten. In this case, the status is 0.

17.12.3 Delete tool environment

DELTOOLENV function

This function can be used to delete sets of data used to describe tool environments. Deletion means that the set of data stored under a particular name can no longer be accessed (an access attempt triggers an alarm).

The DELTOOLENV function is a predefined subprogram.

It must, therefore, be programmed in a separate block.

Syntax:

There are two call formats:

Status = DELTOOLENV()

Status = DELTOOLENV(_NAME)

Value/parameter:

Status INT

0: Function OK

-1: No memory reserved for tool environments:

MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).

i.e. the "tool environments" functionality is not available.

-2: A tool environment with the specified name does not exist.

_NAME STRING

Name of data set to be deleted

The first call format deletes all data sets.

The second call format deletes the data set with the specified name.

Data sets can only be deleted using the DELTOOLENV command, by an INITIAL.INI download or by a cold start (NCK powerup with default machine data). There are no further automatic deletion operations (e.g., on RESET).

17.12.4 How many environments and which ones are saved?

$P_TOOLENVN

This system variable returns the number of available data sets for describing tool environments. (Data sets defined by TOOLENV and not yet deleted.)

The value range is from 0 to machine data:

MD18116 $MN_MM_NUM_TOOL_ENV (number of tool environments in TO area).

This system variable can be accessed even if no tool environments are possible (MD18116 = 0). In this case, the return value is 0.

Syntax:

_N = $P_TOOLENVN

Data type:

_N INT

Number of defined TOOLENV

$P_TOOLENV

This system variable returns the number of the nth data set for describing a tool environment.

The assignment of numbers to data sets is not fixed, but can be changed as a result of deleting or creating data sets. The data sets are numbered internally. The range is from 1 to $P_TOOLENVN.

Syntax:

_NAME = $P_TOOLENV[i]

Data type:

_NAME STRING

Name of the data set with number i

i INT

Number of the data set.

If an index is specified, which does not point to a defined data set, the following alarm is output:

Alarm "17020 (inadmissible array-index1)"

17.12.5 Read T, D, DL from a tool environment

GETTENV function

The GETTENV function is used to read the T, D and DL numbers stored in a tool environment.

The GETTENV function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Syntax:

Status = GETTENV(_NAME, _TDDL)

Value/parameter:

Status INT

0: Function OK

-1: No memory reserved for tool environments:
MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).
i.e. the "tool environments" functionality is not available.

-2: A tool environment with the name specified in **_NAME** does not exist.

_NAME STRING

Name of the tool environment, from which the T, D and DL numbers can be read

_TDDL[3] INT

This integer array contains:
- in "_TDDL[0]" the T number of the tool,
- in "_TDDL[1]" the D number of the tool,
- in "_TDDL[2]" the DL number of the tool,

whose tool environment in the data set is stored with the name "_NAME".

It is possible to omit the first parameter in the GETTENV function call (e.g., GETTENV(, _TDDL)) or to pass a null string as the first parameter (e.g., GETTENV("", _TDDL)). In both of these two special cases, the T, D and DL numbers of the **active** tool are returned in _TDDL.

17.12.6 Read tool lengths, tool length components

GETTCOR function

The GETTCOR function is used to read out tool lengths or tool length components.

The parameters can be used to specify, which components are considered, and the conditions, under which the tool is used.

The GETTCOR function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Syntax:

Status = GETTCOR(_LEN, _COMP, _STAT, _T, _D, _DL)

All parameters can be omitted with the exception of the first parameter (**_LEN**).

Value/parameter:

Status INT

0: Function OK

-1: No memory reserved for tool environments:
MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).
i.e. the "tool environments" functionality is not available.

-2: A tool environment with the name specified in **_STAT** does not exist.

-3: Invalid string in parameter **_COMP**.
Causes of this error can be invalid characters or characters programmed twice.

-4: Invalid T number

-5: Invalid D number

-6: Invalid DL number

-7: Attempt to access non-existent memory module

-8: Attempt to access a non-existent option (programmable tool orientation, tool management).

-9: The **_COMP** string contains a colon (identifier for the specification of a coordinate system), but it is not followed by a valid character denoting the coordinate system.

_LEN[11] REAL

Result vector

The vector components are arranged in the following order:

Tool type	(LEN[0])
Length of cutting edge	(LEN[1])
Abscissa	(LEN[2])
Ordinate	(LEN[3])
Applicate	(LEN[4])
Tool radius	(LEN[5])

The coordinate system defined in **_COMP** and **_STAT** is used as the reference coordinate system for the length components. If no coordinate system is defined in **_COMP**, the tool lengths are represented in the machine coordinate system.

The assignment of the abscissa, ordinate and applicate to the geometry axes depends on the active plane in the tool environment, i.e. with G17, the abscissa is parallel to X, with G18 it is parallel to Z, etc.

Components **LEN[6]** to **LEN[10]** contain the additional parameters, which can be used to specify the geometry description of a tool (e.g. $TC_DP7 to $TC_DP11 for the geometry and the corresponding components for wear or sum and setup offsets).

These 5 additional elements and the tool radius are only defined for components E, G, S, and W. Their evaluation does not depend on **_STAT**. The corresponding values in **LEN[5]** to **LEN[10]** can thus only be not equal to zero if at least one of the four specified components is involved in the tool length calculation. The remaining components do not influence the result. The dimensions refer to the control's basic system (inch or metric).

_COMP STRING

This string consists of two substrings, which are separated from one another by a **colon**.

The individual characters (letters) of the **first substring** identify the tool length components to be taken into account when calculating the tool length.

The **second substring** identifies the coordinate system, in which the tool length is to be output. It consists of only one single relevant character.

The order of the characters in the strings, and their notation (upper or lower case), is arbitrary. Any number of blanks or white spaces can be inserted between the characters.

The letters in the substrings **cannot** be programmed twice. The meanings in the **first substring** are as follows:

-: (Minus symbol, only allowed as first character): The complete tool length is calculated, minus the components specified in the next string.

C: Adapter or tool base dimension (whichever of the two alternative components is active for the tool in use)

E: Setup offsets

G: Geometry

K: Kinematic transformation (is only evaluated for generic 3, 4 and 5-axis transformation)

S: Sum offsets

T: Toolholder with orientation capability

W: Wear

If the first substring is empty (except for white spaces), the complete tool length is calculated allowing for all components. This applies even if the **_COMP** parameter is not specified.

An optional programmable colon must be followed by a single character specifying the coordinate system, in which the tool length components are to be evaluated. If no coordinate system is specified, the evaluation is performed in the MCS (machine coordinate system). If any rotations are to be taken into account, they are specified in the tool environment defined in **_STAT**.

The characters have the following significance:

B: Basic coordinate system (BCS)

K: Tool coordinate system of kinematic transformation (KCS)

M: Machine coordinate system (MCS)

T: Tool coordinate system (TCS)

W: Workpiece coordinate system (WCS)

_STAT STRING

Name of the data set for describing a tool environment.

If the value of this parameter is the null string ("") or is not specified, the current status is used.

_T INT

Internal T number of tool

If this parameter is not specified, or if its value is 0, the tool stored in **_STAT** is used.

If the value of this parameter is -1, the T number of the active tool is used. It is also possible to specify the number of the active tool explicitly.

Note

If **_STAT** is not specified, the current status is used as the tool environment. Since **_T = 0** refers to the T number saved in the tool environment, the active tool is used in that environment, i.e. parameters **_T = 0** and **_T = -1** have the same meaning in this special case.

_D INT

Cutting edge of the tool. If this parameter is not specified, or if its value is 0, the D number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read.

_DL INT

Number of the local compensation. If this parameter is not specified, the DL number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read.

Any rotations and component exchanges initiated by the adapter transformation, toolholder with orientation capability and kinematic transformation, are part of the tool environment. They are thus always performed, even if the corresponding length component is not supposed to be included. If this is undesirable, tool environments must be defined, in which the corresponding transformations are not active. In many cases (i.e. any time a transformation or toolholder with orientation capability is not used on a machine), the data sets stored for the tool environments automatically fulfill these conditions, with the result that the user does not need to make special provision.

MD20360 $MC_TOOL_PARAMETER_DEF_MASK

The two least significant bits of this machine data specify how the wear (bit 0) and tool length (bit 1) are to be evaluated if a diameter axis is used for turning and grinding tools.

If the bits are set, the associated entry is weighted with the factor 0.5. This weighting is reflected in the tool length returned by GETTCOR.

Example:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK = 3 (definition of tool parameters).

MD20100 $MC_DIAMETER_AX_DEF="X" (Geometry axis with face axis funtion)

X is diameter axis (standard turning machine configuration):

```
N30         $TC_DP1[1.1] =      500
N40         $TC_DP2[1.1] =      2
N50         $TC_DP3[1.1] =      3.0          ; Geometry L1
N60         $TC_DP4[1,1]=       4.0
N70         $TC_DP5[1,1]=       5.0
N80         $TC_DP12[1,1]=      12.0         ; Wear L1
N90         $TC_DP13[1,1]=      13.0
N100        $TC_DP14[1,1]=      14.0
N110        t1 d1 g18
N120        r1 = GETTCOR(_LEN, "GW")
N130        r3 = _LEN[2]                     ; 17.0 (= 4.0 + 13.0)
N140        r4 = _LEN[3]                     ; 7.5 (= 0.5 * 3.0 + 0.5 * 12.0)
N150        r5 = _LEN[4]                     ; 19.0 (= 5.0 + 14.0)
N160        m30
```

Kinematic transformation, toolholder with orientation capability

If a toolholder with orientation capability is taken account of during the tool length calculation, the following vectors are included in that calculation:

Type	Vectors
M	l_1 and l_2
T	l_1, l_2 and l_3
P	Tool length is not influenced by the toolholder with orientation capability.

In generic **5-axis transformation**, the following machine data are included in the tool length calculation for transformer types 24 and 56:

Transformer type	Machine data
24	MD24550/24650 $MC_TRAFO5_BASE_TOOL_1/2 MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2 MD24558/24658 $MC_TRAFO5_PART_OFFSET_1/2
56	MD24550/24650 $MC_TRAFO5_BASE_TOOL_1/2 MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2

Transformation type 56 corresponds to type M for a toolholder with orientation capability.

With this 5-axis transformation in the software versions used up to now, the following vector is equivalent to the sum of the two vectors l_1 and l_3 for a toolholder with orientation capability type M.

MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2 (vector of kinematic offset of the first/second 5-axis transformation in the channel)

Only the sum is relevant for the transformation in both cases. The way, in which the two individual components are composed, is insignificant. When calculating the tool length, however, it is relevant which component is assigned to the tool and which is assigned to the tool table.

This explains the introduction of new machine data:

MD24558/24658 $MC_TRAFO5_JOINT_OFFSET_PART_1/2 (vector kinematic offset in table).

It is equivalent to the vector l_3.

The following machine data no longer corresponds to the sum of l_1 and l_3, but only to vector l_1.

MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2 (vector of kinematic offset of the first 5-axis transf. in the channel).

The new response is identical to the current response, if the following machine data equals zero:

MD24558/24658 $MC_TRAFO5_JOINT_OFFSET_PART_1/2 (vector kinematic offset in table).

GETTCOR examples

GETTCOR(_LEN):	Calculates the tool length of the currently active tool in the machine coordinate system allowing for all components.
GETTCOR(_LEN; "CGW : W"):	Calculates the tool length for the active tool, consisting of the adapter or tool base dimension, geometry and wear. Further components, such as toolholder with orientation capability or kinematic transformation, are not considered. The workpiece coordinate system is used for the output.
GETTCOR(_LEN, "-K :B"):	Calculates the complete tool length of the active tool without allowing for the length components of an active kinematic transformation. Output in the basic coordinate system.
GETTCOR(_LEN, ":M", "Testenv1",,3):	Calculates the complete tool length in the machine coordinate system for the tool stored in the tool environment named "Testenv1". The calculation is performed for cutting edge number D3, regardless of the cutting edge number stored.

Compatibility

The GETTCOR function is used in conjunction with the TOOLENV and SETTCOR functions to replace parts of the functionality, which were previously implemented externally in the measuring cycles.

Only some of the parameters, which actually determine the effective tool length, were implemented in the measuring cycles. The above functions can be configured to reproduce the behavior of the measuring cycles in relation to the tool length calculation.

17.12.7 Changing tool components

SETTCOR function

The SETTCOR function is used to change tool components taking into account all general conditions that can be involved when evaluating the individual components.

The SETTCOR function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Note

Regarding the terminology: If in the following, in conjunction with the tool length, tool components are involved, then the components considered from a vectorial perspective are meant, which make up the complete tool length, e.g. geometry or wear. Such a component comprises three individual values (L1, L2, L3), which are called coordinate values in the following.

The tool component "geometry" therefore comprises three coordinate values $TC_DP3 to $TC_DP5.

Syntax:

Status = SETTCOR(_CORVAL, _COMP, _CORCOMP, _CORMODE, _GEOAX, _STAT, _T, _D, _DL)

With the exception of the first two parameters (**_CORVAL** and **_COMP**) all of the parameters can also be omitted.

Value/parameter:

Status INT

0: Function OK

-1: No memory reserved for tool environments:

MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).

i.e. the "tool environments" functionality is not available.

-2: A tool environment with the name specified in **_STAT** does not exist.

-3: Invalid string in parameter **_COMP**.

Causes of this error can be invalid characters or characters programmed twice.

-4: Invalid T number.

-5: Invalid D number.

-6: Invalid DL number.

-7: Attempt to access a non-existent memory module.

-8: Attempt to access a non-existent option (programmable tool orientation, tool management).

-9: Illegal numerical value for the **_CORCOMP** parameter.

-10: Illegal numerical value for the **_CORMODE** parameter.

-11: The contents of the **_COMP** and **_CORRCOMP** parameters are contradictory.

-12: The contents of the **_COMP** and **_CORRMODE** parameters are contradictory.

-13: The content of the **_GEOAX** parameter does not designate a geometry axis.

-14: Write attempt to a non-existent setup offset.

_CORVAL[3] REAL array

Designates the offset vector.

In this case, in the workpiece coordinate system (WCS) defined using **_STAT** the following are assigned:

- _CORVAL[0] of the abscissa
- _CORVAL[1] of the ordinate
- _CORVAL[2] of the applicate

If only one tool component is to be corrected (i.e. no vectorial correction, see parameter **_CORMODE**), the correction value is always in **_CORVAL[0]**, independent of the axis on which it acts. The contents of the other two components are then not evaluated.

If **_CORVAL** or a component of **_CORVAL** refers to the transverse axis, then the data is evaluated as **radius dimension**. This means that a tool is e.g. "longer" by the specified dimension; this correspondingly results in a change to the workpiece diameter that is twice as large.

The dimensions refer to the **basic system** (inch or metric) of the control system.

_COMP STRING

String that comprises either one or two characters. The first or only character for the 1st component (Val_1) and the second character for the 2nd component (Val_2), which are processed according to the subsequent parameters **_CORCOMP** and **_CORMODE**.

The notation of the characters in the string (upper or lower case) is arbitrary. Any number of spaces or tabs (white spaces) can be inserted.

The individual meanings are as follows:

C: Adapter or tool base dimension (whichever of the two alternative components is active for the tool in use)

E: Setup offsets

G: Geometry

S: Sum offsets

W: Wear

_CORCOMP INT

This parameter specifies the component(s) of the two data sets that are to be described. If this parameter is not specified then its value is 0.

Meaning of the numerical values:

0: The offset value _CORVAL[0] refers to the geometry axis transferred in parameter _GEOAX in the workpiece coordinate system, i.e. the offset value must be calculated in the designated tool components so that, taking account all the parameters that can influence the tool length calculation, as a result, a change of the total tool length by the specified value in the specified axis direction is obtained.

This change should be achieved by the correction of the component specified in _COMP and the symbolic algorithm specified in _CORMODE (see the following parameters). The resulting correction can therefore have an effect on all three axis components.

1: Like 0, however, vectorial. The content of vector _CORVAL refers to abscissa, ordinate, applicate in the workpiece coordinate system (WCS).

The subsequent parameter _GEOAX is not evaluated.

2: Vectorial offset, i.e. L1, L2 and L3 can change simultaneously.

In contrast to the versions from 0 and 1, the offset values contained in _CORVAL refer to the coordinates of Val_1 components (see following parameter _CORMODE) of the tool.

Any possible inclination of an existing tool compared with the workpiece coordinate system has no influence on the offset.

3 - 5: Correction of tool lengths L1 to L3 ($TC_DP3 to $TC_DP5) or the corresponding values for wear, setting up or additive offsets.

The offset value is contained in _CORVAL[0]. It is measured in the coordinates of the Val_1 component (see following parameter _CORMODE) of the tool. Any possible inclination of an existing tool compared with the workpiece coordinate system has no influence on the offset.

6: Correction of the tool radius ($TC_DP6) or the corresponding values for wear, setting up or additive offsets.

7 – 11: Correction of $TC_DP7 to $TC_DP11 or the corresponding values for wear, setting up or additive offsets. These parameters are treated just like the tool radius.

_CORMODE INT

This parameter specifies the type of write operation to be executed.

If this parameter is not specified then its value is 0.

Meaning of the ones location:

0: Val_{1new} = _CORVAL

1: Val_{1new} = Val_{1old} + _CORVAL

2: Val_{1new} = _CORVAL

Val_{2new} = 0

3: Val_{1new} = Val_{1old} + Val_{2old} + _CORVAL

Val_{2new} = 0

The notation Val_{1old} + Val_{2old} is symbolic. If the two components (due to the status of **_STAT**) are evaluated in different ways, i.e. if a rotation is effective between the two components, then Val_{2old} is transformed prior to addition so that the resulting tool length after deleting Val_{2new} and prior to the addition of **_CORVAL** remains unchanged.

_CORVAL always refers to Val_1. **_CORVAL** is a value that is always measured in the workpiece coordinate system (WCS). It is therefore already transformed with respect to the tool components, in which it should be calculated. Therefore, it cannot be directly calculated together with the saved value, but must be transformed back prior to adding to Val_1 or Val_2.

This can mean that the offset acts on an axis different than the one defined by **_CORCOMP** – or that it acts on several axes.

For the case **CORRCOMP = 0**, i.e. if **_CORVAL** does not contain a vector, but only an individual value, then the described operations are executed in the coordinates in which **_CORVAL** was measured (WCS). In particular, this also applies to setting Val_{2new} to zero in versions 2 and 3. This result is then transformed back into the coordinates of the tool. This can mean that none of the coordinate values to be set to zero (L1, L2, L3) become zero, or coordinate values, that were previously zero, are now not equal to zero. However, if the corresponding operations are successively executed for all three geometry axes, then all three coordinate values of the components to be deleted are always zero. If the tool is not rotated with respect to the workpiece coordinate system or is rotated so that all tool components remain parallel to the coordinate axes (axis exchange operations), then this also ensures that only one tool coordinate changes.

The successive execution of the same operation (**_CORRMODE**) with **_CORCOMP = 0** for all three coordinate axes in any sequence is identical with the single execution of the same operation with **_CORCOMP = 1**.

_T INT

Internal T number of the tool. If this parameter is not specified or if its value is 0, then the tool stored in **_STAT** is used. If the value of this parameter is -1, the T number of the active tool is used. It is also possible to explicitly specify the number of the active tool.

Note

If **_STAT** is not specified, the actual status is used as the tool environment. Since **_T = 0** refers to the T number saved in the tool environment, the active tool is used in this environment, i.e. parameters **_T = 0** and **_T = -1** have the same meaning in this special case.

_D INT

Cutting edge of the tool. If this parameter is not specified, or if its value is 0, the D number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read.

_DL INT

Number of the offset dependent on the location. If this parameter is not specified, the DL number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read. If T, D and DL specify a tool without location-dependent offsets, no additive or setting-up offsets may be specified in parameter **_COMP** (error code in "Status").

Not all possible combinations of the three parameters **_COMP**, **_CORCOMP** and **_CORMODE** are sensible. For example, algorithm 3 in **_CORCOMP** requires that two characters are specified in **_COMP**. If an invalid parameter combination is specified, then a corresponding error code is returned in the status.

Note

Calculating the tool length depending on machine data MD20360

The two least significant bits of this machine data specify how the wear (bit 0) and tool length (bit 1) are to be evaluated if a diameter axis is used for turning and grinding tools. If the appropriate bits are set, then for the tool length calculation, a factor of 0.5 is applied to the associated entry. The correction using `SETTCOR` is executed so that the total effective tool length change is equal to the value transferred in **_CORVAL**.

The correction of the components, whose length is evaluated with a factor of 0.5 due to the following machine data for the length calculation, must be realized using twice the transferred value:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters)

Example 1

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] =          120           ; Milling tool
N30     $TC_DP3[1.1] =          10.0          ; Geometry L1
N40     $TC_DP12[1,1]=          1.0           ; Wear L1
N50     _CORVAL[0] =            0.333
N60     t1 d1 g17 g0
N70     r1 = settcor(_CORVAL, "G", 0, 0, 2)
N80     t1 d1 x0 y0 z0                        ; ==>    MCS position X0.000 Y0.000
                                                       Z1.333
N90     M30
```

_CORCOMP is 0, therefore, the coordinate value of the geometry component acting in the Z direction must be replaced by the offset value 0.333. The resulting total tool length is thus L1 = 0.333 + 1.000 = 1.333

Example 2

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] =          120           ; Milling tool
N30     $TC_DP3[1.1] =          10.0          ; Geometry L1
N40     $TC_DP12[1,1]=          1.0           ; Wear L1
N50     _CORVAL[0] =            0.333
N60     t1 d1 g17 g0
N70     r1 = settcor(_CORVAL, "W", 0, 1, 2)
N80     t1 d1 x0 y0 z0                        ; ==>    MCS position X0.000 Y0.000
                                                       Z11.333
N90     M30
```

_CORCOMP is 1, therefore, an offset value of 0.333 acting in the Z direction is added to the wear value of 1.0.

Therefore, the resulting total tool length is L1 = 10.0 + 1.333 = 11.333.

17.13 Tool lengths L1, L2, L3 assignment: LENTOAX

LENTOAX function

The "LENTOAX" function provides information about the assignment of tool lengths L1, L2 and L3 of the **active** tool to the abscissa, ordinate and applicate. The assignment of abscissa, ordinate and applicate to the geometry axes is affected by frames and the active plane (G17 - G19).

Only the geometry component of a tool ($TC_DP3[x,y] to $TC_DP5[x,y]) is considered, i.e. a different axis assignment for other components (e.g. wear) has no effect on the result.

The "LENTOAX" function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Syntax:

Status = LENTOAX(_AXIND, _MATRIX, _COORD)

The first two parameters must always be programmed; the last parameter can be omitted.

Value/parameter:

Status INT

0: Function OK, information in **_AXIND** sufficient for description (all tool length components are parallel to the geometry axes).

1: Function is OK, however, the content of **_MATRIX** must be evaluated for a correct description (the tool length components are not parallel to the geometry axes).

-1: Invalid string in parameter **_COORD**.

-2: No tool active.

_AXIND[3] INT array

Indices 0 to 2 are assigned to the abscissa (0), ordinate (1) and applicate (2) (e.g. **_AXIND[0]** contains the number of the tool length components, which are effective in the direction of the abscissa).

The content has the following significance:

0: Assignment exists (axis does not exist)

1 to 3: or **-1 to -3:** Number of the length effective in the corresponding coordinate axis. The sign is negative if the tool length component is pointing in the negative coordinate direction.

_MATRIX[3][3] REAL array

Matrix which represents the vector of the tool lengths (L1=1, L2=1, L3=1) to the vector of the coordinate axes (abscissa, ordinate, applicate), i.e. the tool length components are assigned to the **columns** in the order L1, L2, L3 and the axes are assigned to the **lines** in the order abscissa, ordinate, applicate.

All elements are always valid in the matrix, even if the geometry axis belonging to the coordinate axis is not available, i.e. if the corresponding entry in **_AXIND** is 0.

_COORD STRING

Specifies the coordinate system used for the assignment.

MCS or **M**:	The tool length is represented in the machine coordinate system.
BCS or **B**:	The tool length is represented in the basic coordinate system.
WCS or **W**:	The tool length is represented in the workpiece coordinate system (default).
KCS or **K**:	The tool length is represented in the tool coordinate system of the **kinematic transformation**.
TCS or **T**:	The tool length is represented in the tool coordinate system.

The notation of the characters in the string (upper or lower case) is arbitrary.

Further explanations

If the tool length components are parallel to the geometry axes, the axis indices assigned to length components L1 to L3 are returned in the **_AXIND** array.

If a tool length component points in the negative axis direction, the associated axis index contains a minus sign. In this case, the return value (**status**) is 0. If an axis does not exist, the associated return value is 0. The assignment can also be read from the **_MATRIX** parameter. Six of the nine matrix elements are then zero, and three elements contain the value +1 or -1.

Note

In the TCS, all tool length components are always parallel or antiparallel to the axes.

The components can only be antiparallel when mirroring is active and the following setting data is activated:

SD42900 $SC_MIRROR_TOOL_LENGTH (sign change tool length when mirroring)

If not all length components are parallel or antiparallel to the geometry axes, the index of the axis, which contains the largest part of a tool length component, is returned in _AXIND. In this case (if the function does not return an error for a different reason), the return value is 1. The mapping of tool length components L1 to L3 onto geometry axes 1 to 3 is then described completely by the contents of the 3rd parameter **_MATRIX**.

The **_COORD** parameter can be used to specify, which coordinate system is to be used for the geometry axes. If the **_COORD** parameter is not specified (notation **LENTOAX(_AXIND, _MATRIX)**), the WCS is used (default).

Example:

Standard situation: milling tool with G17

L1 applies in Z (applicate), L2 applies in Y (ordinate), L3 applies in X (abscissa).

Function call in the form:

Status = LENTOAX(_AXIND, _MATRIX, "WCS")

The result parameter **_AXIND** contains the values:

_AXIND[0] = 3

_AXIND[1] = 2

_AXIND[2] = 1

Or, in short: (3, 2, 1)

In this case, the associated matrix **_MATRIX** is:

$$\_MATRIX = \begin{pmatrix} & & 1 \\ & 1 & \\ 1 & & \end{pmatrix}$$

A change from G17 to G18 or G19 does not alter the result, because the assignment of the length components to the geometry axes changes in the same way as the assignment of the abscissa, ordinate and applicate.

A frame rotation of Z through 60 degrees is now programmed with `G17` active, e.g. `rot Z60`. The direction of the applicate (Z direction) remains unchanged; the main component of `L2` now lies in the direction of the new X axis; the main component of `L1` now lies in the direction of the negative Y axis. The return axis is thus 1, and **_AXIND** contains the values (2, -3, 1).

In this case, the associated matrix **_MATRIX** is:

$$\_MATRIX = \begin{pmatrix} \cos(\quad) & \sin(\quad) & \\ \sin(\quad) & \cos(\quad) & \\ & & 1 \end{pmatrix}$$

Note

For further information to the above mentioned coordinate systems, please refer to:

References:
Programming Manual, Job Planning; Tool Offsets:

17.14 Supplementary conditions

17.14.1 Flat D number structure

Grinding tools

Grinding tools (tool types 400-499) cannot be defined using the simple tool management structure (flat D numbers).

Block search

T number output to PLC triggers a synchronization process in the NCK: with absolute, indirect D programming, the PLC returns the D values via NC/PLC interfaces. The NCK waits until the output of a T number is followed by a response from the PLC: "I have written the D number". With block search without calculation, this process of synchronization must be deactivated until the first valid T number has been output again. That means that the NCK must not wait on D programming.

Note

At what point the auxiliary functions can be output to PLC after block search is complete, can be controlled with the machine data:

$MC_AUXFU_AT_BLOCK_SEARCH_END (auxiliary function output after block search)

Automatic on end or on NC start.

REORG

The (only) writable variable $A_MONIFACT, which is defined here, is stored by main-run data. Since the write process takes place synchronously to the main run, no special measures are required for Reorg.

17.14.2 SD42935 expansions

SD42935

Which of the wear components are to be transformed and which are not to be transformed in conjunction with the functions `TOWMCS` and `TOWWCS` can be defined via the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformation of wear values)

17.15 Examples

17.15.1 Toolholder with orientation capability

17.15.1.1 Example: Toolholder with orientation capability

Requirement

The following example uses a toolholder, which is described fully by a rotation about the Y axis. It is therefore sufficient to enter only one value to define the rotary axis (block N20).

Blocks N50 to N70 describe an end mill with radius 5 mm and length 20 mm.

Block N90 defines a rotation of 37 degrees about the Y axis.

Block N120 activates the tool radius compensation and all settings are made to describe the compensation in the following blocks with a rotation of 37 degrees about the Y axis.

```
N10                                    ; Definition of toolholder 1
N20 $TC_CARR8[1] = 1                   ; Component of the first rotary axis in
                                       the Y direction
N30
N40                                    ; Definition of tool-compensation memory
                                       T1/D1
N50 $TC_DP1[1,1] = 120                 ; End mill
N60 $TC_DP3[1,1] = 20                  ; Length 1
N70 $TC_DP6[1,1] = 5                   ; Radius
N80
N90 ROT Y37                            ; 37-degree rotation about y axis
N100
N110 X0 Y0 Z0 F10000
N120 G42 CUT2DF TCOFR TCARR = 1 T1 D1 X10
N130 X40
N140 Y40
N150 X0
N160 Y0
N170 M30
```

17.15.1.2 Example of toolholder with orientation capability with rotary table

Use of the MOV command

For use of the MOVT command it is assumed that the program is running on a 5axis machine, on which the tool rotates about the Y axis in case of a rotation of the B axis:

```
N10 TRAORI()
N20 X0 X0 Z0 B45 F2000                    ; Setting the tool orientation
N30 MOVT=-10                              ; Infeed movement 10 mm in tool
                                          direction
                                          ; (under 45 degrees in the Y-Z plane)
N40 MOVT=AC(20)                           ; Retraction in tool direction at
                                          distance of
                                          ; 20 mm from the zero point
```

Machine with rotary table

Complete definition for the use of a toolholder with orientation capability with rotary table:

```
N10 $TC_DP1[1,1]=120
N20 $TC_DP3[1,1]= 13                    ; Tool length 13 mm

; Definition of toolholder 1:

N30 $TC_CARR1[1] = 0                    ; X component of 1st offset vector
N40 $TC_CARR2[1] = 0                    ; Y component of 1st offset vector
N50 $TC_CARR3[1] = 0                    ; Z component of 1st offset vector

N60 $TC_CARR4[1] = 0                    ; X component of 2nd offset vector
N70 $TC_CARR5[1] = 0                    ; Y component of 2nd offset vector
N80 $TC_CARR6[1] = -15                  ; Z component of 2nd offset vector

N90 $TC_CARR7[1] = 1                    ; X component of 1st axis
N100 $TC_CARR8[1] = 0                   ; Y component of 1st axis
N110 $TC_CARR9[1] = 0                   ; Z component of 1st axis

N120 $TC_CARR10[1] = 0                  ; X component of 2nd axis
N130 $TC_CARR11[1] = 1                  ; Y component of 2nd axis
N140 $TC_CARR12[1] = 0                  ; Z component of 2nd axis

N150 $TC_CARR13[1] = 30                 ; Angle of rotation of 1st axis
N160 $TC_CARR14[1] =-30                 ; Angle of rotation of 2nd axis

N170 $TC_CARR15[1] = 0                  ; X components of 3rd offset vector
```

```
N180 $TC_CARR16[1] = 0                    ; Y component of 3rd offset vector
N190 $TC_CARR17[1] = 0                    ; Z component of 3rd offset vector

N200 $TC_CARR18[1] = 0                    ; X component of 4th offset vector
N210 $TC_CARR19[1] = 0                    ; Y component of 4th offset vector
N220 $TC_CARR20[1] = 15                   ; Z component of 4th offset vector

N230 $TC_CARR21[1] = A                    ; Reference for 1st axis
N240 $TC_CARR22[1] = B                    ; Reference for 2nd axis
N250 $TC_CARR23[1] = "P"                  ; Toolholder type

N260 X0 Y0 Z0 A0 B45 F2000
N270 TCARR=1 X0 Y10 Z0 T1 TCOABS
N280 PAROT
N290 X0 Y0 Z0
N300 G18 MOVT=AC(20)
N310 G17 X10 Y0 Z0
N320 MOVT=-10
N330 PAROTOF
N340 TCOFR
N350 X10 Y10 Z-13 A0 B0
N360 ROTS X-45 Y45
N370 X20 Y0 Z0 D0
N380 Y20
N390 X0 Y0 Z20
N400 M30
```

The definition of the toolholder with orientation capability is given in full. The components which contain the value 0 need not actually be given, as they are preset to zero in any case.

The toolholder is activated in N270.

As *$TC_CARR21* and *$TC_CARR22* refer to the machine axes A and B and TCOABS is active, the values in *$TC_CARR13* and *$TC_CARR14* are ignored, i.e. the axis position A0 B45 is used for the rotation.

The rotation of the 4th offset vector (length 15 mm in Z direction) around the B axis causes an offsetting of the zero point by X10.607 [= 15 * sin(45)] and Z-4.393 [= -15 * (1. - cos(45))]. This zero offset is taken into account by an automatically written basic or system frame so that the position X10.607 Y10.000 Z8.607 is approached. In the Z direction the tool selection leads to an additional offset of 13 mm; the Y component is not affected by the table rotation.

N280 defines a rotation in accordance with the rotation of the table of the toolholder with orientation capability. The new X direction thus points in the direction of the bisecting line in the 4th quadrant, the new Z axis in the direction of the bisecting line in the 1st quadrant.

The zero point is approached in N290, i.e. the machine position X10.607 Y0 Z-4.393, since the position of the zero point is not changed by the rotation.

N300 traverses in Y to the position Y33.000, since G18 is active and the Y component is not affected by the active frame. The X and Z positions remain unchanged.

The position X17.678 Y0 Z1.536 is approached in N310.

N320 changes only the Z position to the value -8.464 as a result of the MOVT command. As only the table can be rotated, the tool orientation remains unchanged parallel to the machine Z direction, even if the Z direction of the active frame is rotated by 45 degrees.

N330 deletes the basic or system frame; thus the frame definition from N280 is undone.

In N340, TCOFR specifies that the toolholder with orientation capability is to be aligned according to the active frame. Since a rotation is no longer active in N330 due to the PAROTOF command, the initial state is applied. The frame offset becomes zero.

N350 thus approaches the position X10 X10 Z0 (= Z-13 + tool length). Note: Through the simultaneous programming of both rotary axes A and B the actual position of the toolholder with orientation capability is made to match that used in N340. The position approached by the three linear axes is dependent on this position, however.

In N360, solid angles are used to define a plane whose intersecting lines in the XZ and in the YZ plane each form an angle of +45 degrees or -45 degrees with the X or Y axis. The plane defined in such a way therefore has the following position: the surface normal points towards the solid diagonals.

N370 traverses to the position X20 Y0 Z0 in the new coordinate system. Since the tool is deselected with D0 at the same time, there is no longer an additional offset in Z. Since the new X axis lies in the old XZ plane, this block reaches the machine position X14.142 Y0 Z-14.142.

N380 only traverses on the Y axis in the rotated coordinate system. This leads to a motion of all three machine axes. The machine position is X5.977 Y16.330 Z-22.307.

N390 approaches a point on the new Z axis. Relative to the machine axes this is thus on the solid diagonal. All three axes thus reach the position 11.547.

17.15.1.3 Basic tool orientation example

Basic orientation in the bisector

A milling tool is defined with length L1=10 whose basic orientation is in the bisector of the XZ plane.

```
N10     $TC_DP1[1,1]=120
N20     $TC_DP3[1,1]=10
N30     $TC_DPV [1,1] = 0
N40     $TC_DPV3[1,1] = 1
N50     $TC_DPV4[1,1] = 0
N60     $TC_DPV5[1,1] = 1
N70     g17 f1000 x0 y0 z0 t1 d1
N80     movt=10
N80     m30
```

Description of example:

In N10 to N60, a milling tool is defined with length L1=10 (N20). The basic orientation is in the bisector of the XZ plane N40 to N60.

In N70, the tool is activated and the zero position is approached. As a result of the tool length the machine positions X0 Y0 Z10 are thus obtained in this block.

In N80 an incremental traversing motion is performed from 10 into tool direction. The resulting axis positions are thus X7.071 Y0 Z17.071.

17.15.1.4 Calculation of compensation values on a location-specific and workpiece-specific basis

Tool with adapter

A tool with adapter and toolholder with orientation capability is defined in the following program example. In order to simplify the overview, only length L1 is different to zero for the additive and insert offsets and for the adapter in case of the tool itself. The offset vectors of the toolholder with orientation capability are all zero.

```
N10     $TC_TP2[1] = "MillingTool"              ; Name of identifier
N20     $TC_TP7[1]=9                            ; Location types
N30     $TC_TP8[1]=2                            ; Status: enabled and not blocked

; D corr. D=1

N40     $TC_DP1[1,1]=120                        ; Tool type - milling
N50     $TC_DP3[1,1]=; tool length compensation
        vector
N60     $TC_DP12[1,1]= ; wear
```

```
N70     $TC_SCP13[1,1]=0.1                      ; Sum offset DL=1
N80     $TC_ECP13[1,1]=0.01                     ; Insert offset DL=1
N90     $TC_ADPTT[1]=5                          ; Adapter transformation
N100    $TC_ADPT1[1]=0.001                      ; Adapter dimension

                                                ; Magazine data
N110    $TC_MAP1[1]=3                           ; Magazine type: Revolver
N120    $TC_MAP2[1]="Revolver"                  ; Magazine identifier
N130    $TC_MAP3[1]=17                          ; Status of magazine
N140    $TC_MAP6[1]=1                           ; Dimension - line
N150    $TC_MAP7[1]=2                           ; Dimension - column -> 2 positions
N160    $TC_MPP1[1,1]=1                         ; Location type
N170    $TC_MPP2[1,1]=9                         ; Location types
N180    $TC_MPP4[1,1]=2                         ; Location state
N190    $TC_MPP7[1,1]=1                         ; Bring adapter into position
N200    $TC_MPP6[1,1]=1                         ; T number "MillingTool"
N210    $TC_MAP1[9999]=7                        ; Magazine type: buffer
N220    $TC_MAP2[9999]="buffer"                 ; Magazine identifier
N230    $TC_MAP3[9999]=17                       ; Status of magazine
N240    $TC_MAP6[9999]=1                        ; Dimension - line
N250    $TC_MAP7[9999]=1                        ; Dimension - column -> 1 position
N260    $TC_MPP1[9999.1]=2                      ; Location type
N270    $TC_MPP2[9999.1]=9                      ; Location types
N280    $TC_MPP4[9999.1]=2                      ; Location state
N290    $TC_MPP5[9999,1]=1                      ; Spindle no. 1
N300    $TC_MDP2[1,1]=0                         ; Distance from spindle to mag. 1

                                                ; Definition of toolholder 1
N310    $TC_CARR10[1] = 1                       ; Component of 2nd rotary axis in X
                                                direction
N320    $TC_CARR14[1] = 45                      ; Angle of rotation of 2nd axis
N330    $TC_CARR23[1] = "T"                     ; Tool mode
N340    Stopre
N350    $SC_WEAR_TRANSFORM = 'B101'
N360    T0 D0 DL=0
N370    ROT X30
N380    G90 G1 G17 F10000 X0 Y0 Z0
N390    T="MillingTool" X0 Y0 Z0 TOWSTD         ; X 0.000 Y11.110 Z 0.001
N400    T="MillingTool" X0 Y0 Z0 TOWMCS         ; X 0.000 Y10.100 Z 1.011
N410    T="MillingTool" X0 Y0 Z0 TOWWCS         ; X 0.000 Y 9.595 Z 0.876
N420    TCARR=1 X0 Y0 Z0                        ; X 0.000 Y 6.636 Z 8.017
N430    G18 X0 Y0 Z0                            ; X10.100 Y-0.504 Z 0.876
N440    m30
```

Explanations regarding the example above

Starting at block N390, various methods are used to approach position X0 Y0 Z0. The machine positions reached are specified in the blocks in comments. After the program a description is given of how the positions were reached.

N390: The adapter transformation 5 (block N90) transforms length L1 into length L2. Only the actual adapter dimension is not subject to this transformation. The Y value (L2 with G17) results from the sum of the tool length (10), tool wear (1), sum offset (0.1), and insert offset (0.01). The adapter dimension (0.001) is in Z (L1).

N400: In block N350, bits 0 and 2 are enabled in setting data:

SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)

This means that the tool wear and the insert offset are not subject to the adapter transformation because of TOWMCS in block N400. The sum of these two compensations is 1.01. The Z position is, therefore, increased by this amount and the Y position is reduced by this amount compared with block N390.

TOWWCS is active in N410. The sum of the tool wear and the insert offset is thus effective in the active workpiece coordinate system. In block N370, a rotation through 30 degrees is activated about the X axis. The original compensation value of 1.01 in the Z direction thus yields a new Z component of 0.875 (= 1.01 * cos(30)) and a new Y component of -0.505 (= 1.01 * sin(30)). This yields the dimension specified in the program comment when added to the sum of the tool length, sum offset and adapter dimension produced in block N390.

In addition, a toolholder with orientation capability is activated in block N420. This executes a rotation through 45 degrees about the X axis (see N310 - N330). Since all offset vectors of the toolholder are zero, there is no additional zero offset. The toolholder with orientation capability acts on the sum of the tool length, sum offset and adapter dimension. The resulting vector component is X0 Y7.141 Z7.142. To this, as in block N410, the sum of tool wear and insert offset evaluated in WCS is added.

G18 is activated in N430. The components of the sum of the tool length, sum offset and adapter dimension are interchanged accordingly. The toolholder with orientation capability continues to act on this new vector (rotation through 45 degrees about X axis). The resulting vector component thereby is X10.100 Y0.0071 Z0.0071. The vector formed from tool wear and insert offset (X0 Y-0.505 Z0.875) is not affected by the change of plane. The sum of the two vectors yields the dimension specified in the comment in N430.

17.15.2 Examples 3-6: SETTCOR function for tool environments

Example 3

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] = 120                      ; Milling tool
N30     $TC_DP3[1,1] = 10.0                     ; Geometry L1
N40     $TC_DP12[1,1] = 1.0                     ; Wear L1
N50     _CORVAL[0] = 0.333
N60     t1 d1 g17 g0
N70     r1 = settcor(_CORVAL, "GW", 0, 2, 2)
N80     t1 d1 x0 y0 z0                          ; ==> MCS position X0.000 Y0.000
                                                Z0.333
N90     M30
```

_CORCOMP is 2, therefore, the compensation effective in the Z direction is entered in the geometry component (the old value is overwritten) and the wear value is deleted. The resulting total tool length is thus:

L1 = 0.333 + 0.0 = 0.333.

Example 4

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] = 120                      ; Milling tool
N30     $TC_DP3[1,1] = 10.0                     ; Geometry L1
N40     $TC_DP12[1,1] = 1.0                     ; Wear L1
N50     _CORVAL[0] = 0.333
N60     t1 d1 g17 g0
N70     r1 = settcor(_CORVAL, "GW", 0, 3, 2)
N80     t1 d1 x0 y0 z0                          ;==> MCS position X0.000 Y0.000
                                                Z11.333
N90     M30
```

_CORCOMP is 3, therefore, the wear value and compensation value are added to the geometry component and the wear component is deleted. The resulting total tool length is thus L1 = 11.333 + 0.0 = 11.333.

Example 5

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] = 120                          ; Milling tool
N30     $TC_DP3[1,1] = 10.0                         ; Geometry L1
N40     $TC_DP12[1,1] = 1.0                         ; Wear L1
N50     _CORVAL[0] = 0.333
N60     t1 d1 g17 g0
N70     r1 = settcor(_CORVAL, "GW", 0, 3, 0)
N80     t1 d1 x0 y0 z0                              ;==> MCS position X0.333 Y0.000
                                                    Z11.000
N90     M30
```

_CORCOMP is 3, as in the previous example, but the compensation is now effective on the geometry axis with index 0 (X axis). The tool components `L3` are assigned to this geometry axis due to `G17` with a milling tool. Calling SETTCOR thus does not affect tool parameters $TC_DP3 and $TC_DP12. Instead, the compensation value is entered in $TC_DP5.

Example 6

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] = 500                                  ; Turning tool
N30     $TC_DP3[1,1] = 10.0                                 ; Geometry L1
N40     $TC_DP4[1,1] = 15.0                                 ; Geometry L2
N50     $TC_DP12[1,1]= 10.0                                 ; Wear L1
N60     $TC_DP13[1,1] =0.0                                  ; Wear L2
N70     _CORVAL[0] = 5.0
N80     rot y 30
N90     t1 d1 g18 g0
N100    r1 = settcor(_CORVAL, "GW", 0, 3, 1)
N110    t1 d1 x0 y0 z0                                      ; ==> MCS position X24.330
                                                            Y0.000 Z17.500
N120    M30
```

The tool is a turning tool. A frame rotation is activated in N80, causing the basic coordinate system (BCS) to be rotated in relation to the workpiece coordinate system (WCS). In the WCS, the compensation value (N70) acts on the geometry axis with index 1, i.e., on the X axis because G18 is active. Since "_CORRMODE = 3", the tool wear in the direction of the X axis of the WCS must become zero once N100 has been executed. The contents of the relevant tool parameters at the end of the program are thus:

```
$TC_DP3[1,1]                : 21.830                    ; Geometry L1
$TC_DP4[1,1]                : 21.830                    ; Geometry L2
$TC_DP12[1,1]               : 2.500                     ; Wear L1
$TC_DP13[1,1]               : -4.330                    ; Wear L2
```

The total wear including _CORVAL is mapped onto the X' direction in the WCS. This produces point P2. The coordinates of this point (measured in X/Y coordinates) are entered in the geometry component of the tool. The difference vector P_2 - P_1 remains in the wear. The wear thus no longer has a component in the direction of _CORVAL.

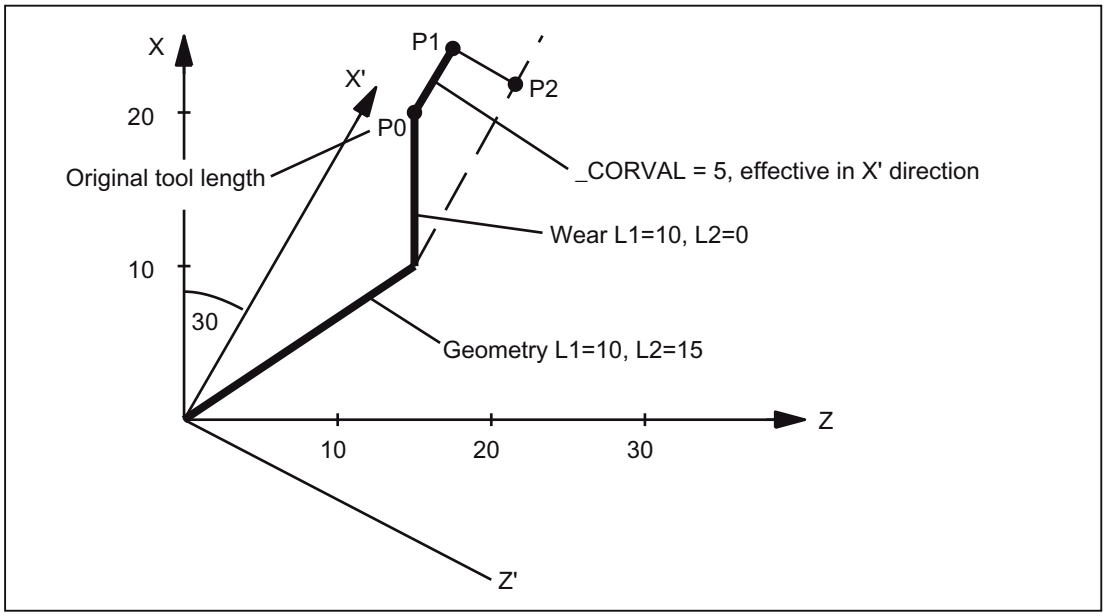


Figure 17-54 Tool length compensation, example 6

If the sample program is continued after N110 with the following instructions:

```
N120    _CORVAL[0] = 0.0
N130    r1 = settcor(_CORVAL, "GW", 0, 3, 0)
N140    t1 d1 x0 y0 z0                          ; ==> MCS position X24.330 Y0.000
                                                Z17.500
```

The remaining wear is included completely in the geometry because the compensation is now effective in the Z' axis (parameter _GEOAX is 0). Since the new compensation value is 0, the total tool length and thus the position approached in N140 may not change. If _CORVAL were not equal to 0 in N120, a new total tool length and thus a new position in N140 would result, however, the wear component of the tool length would always be zero, i.e., the total tool length is subsequently always contained in the geometry component of the tool.

The same result as that achieved by calling the SETTCOR function with the _CORCOMP = 0 parameter twice can also be reached by calling _CORRCOMP = 1 (vectorial compensation) just once:

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] = 500                          ; Turning tool
N30     $TC_DP3[1,1] = 10.0                         ; Geometry L1
N40     $TC_DP4[1,1] = 15.0                         ; Geometry L2
N50     $TC_DP12[1,1]= 10.0                         ; Wear L1
N60     $TC_DP13[1,1] =0.0                          ; Wear L2
N70     _CORVAL[0] = 0.0
N71     _CORVAL[1] = 5.0
N72     _CORVAL[2] = 0.0
N80     rot y 30
N90     t1 d1 g18 g0
N100    r1 = settcor(_CORVAL, "GW", 1, 3, 1)
N110    t1 d1 x0 y0 z0                              ; ==> MCS position X24.330 Y0.000
                                                    Z17.500
N120    M30
```

In this case, all wear components of the tool are set to zero immediately after the first call of SETTCOR in N100.

Example 7

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] = 500                          ; Turning tool
N30     $TC_DP3[1,1] = 10.0                         ; Geometry L1
N40     $TC_DP4[1,1] = 15.0                         ; Geometry L2
N50     $TC_DP12[1,1]= 10.0                         ; Wear L1
N60     $TC_DP13[1,1] =0.0                          ; Wear L2
N70     _CORVAL[0] = 5.0
N80     rot y 30
N90     t1 d1 g18 g0
N100    r1 = settcor(_CORVAL, "GW", 3, 3)
N110    t1 d1 x0 y0 z0                              ; ==> MCS position X25.000 Y0.000
                                                    Z15.000
```

As opposed to example 6, parameter _CORCOMP = 3, and so the _GEOAX parameter can be omitted. The value contained in _CORVAL[0] now acts immediately on the tool length component L1, the rotation in N80 has no effect on the result, the wear components in $TC_DP12 are included in the geometry component together with _CORVAL[0], with the result that the total tool length is stored in the geometry component of the tool, due to $TC_DP13, after the first SETTCOR call in N100.

Example 8

```
N10     def real _CORVAL[3]
N20     $TC_DP1[1,1] = 500                          ; Turning tool
N30     $TC_DP3[1,1] = 10.0                         ; Geometry L1
N40     $TC_DP4[1,1] = 15.0                         ; Geometry L2
N50     $TC_DP5[1,1] = 20.0                         ; Geometry L3
N60     $TC_DP12[1,1]= 10.0                         ; Wear L1
N70     $TC_DP13[1,1] =0.0                          ; Wear L2
N80     $TC_DP14[1.1] =0.0                          ; Wear L3
N90     $SC_WEAR_SIGN = TRUE
N100    _CORVAL[0] = 10.0
N110    _CORVAL[1] = 15.0
N120    _CORVAL[2] = 5.0
N130    rot y 30
N140    t1 d1 g18 g0
N150    r1 = settcor(_CORVAL, "W", 1, 1)
N160    t1 d1 x0 y0 z0                              ; ==> MCS position X7.990 Y25.000
                                                    Z31.160
```

In N90 the setting data is enabled:

SD42930 $SC_WEAR_SIGN (sign of wear)

i.e. the wear must be valued with a negative sign.

The compensation is vectorial (_CORCOMP = 1), and the compensation vector must be added to the wear (_CORMODE = 1). The geometric conditions in the Z/X plane are shown in the figure below:

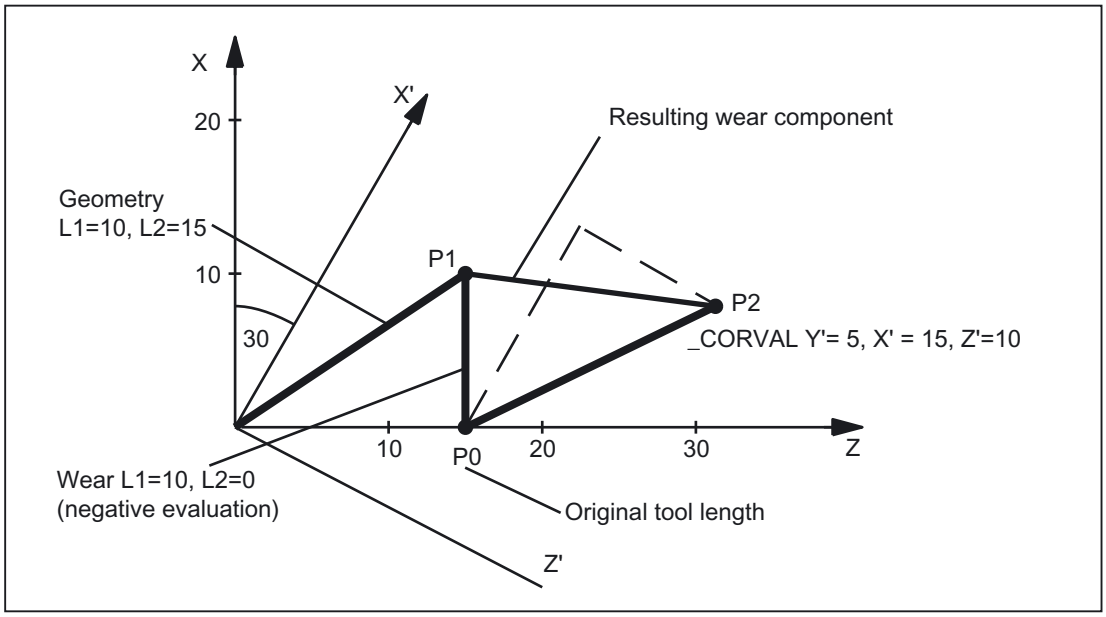


Figure 17-55 Tool length compensation, example 8

The geometry component of the tool remains unchanged due to _CORMODE = 1. The compensation vector defined in the WCS (rotation about y axis) must be included in the wear component such that the total tool length in the figure refers to point P_2. Therefore, the resulting wear component of the tool is given in relation to the distance between points P_1 and P_2.

However, since the wear is evaluated negatively, due to setting data SD42930, the compensation determined in this way has to be entered in the compensation memory with a negative sign. The contents of the relevant tool parameters at the end of the program are thus:

```
$TC_DP3[1,1]          :  10.000                  ; Geometry L1 (unchanged)
$TC_DP4[1,1]          :  15.000                  ; Geometry L2 (unchanged)
$TC_DP5[1,1]          :  10.000                  ; Geometry L3 (unchanged)

$TC_DP12[1,1]         :  2.010                   ; Wear L1
                                                  ;(= 10 -15*cos(30) + 10*sin(30))
$TC_DP13[1,1]            : -16.160               ; Wear L2
                                                  ;(= -15*sin(30) - 10*cos(30))
$TC_DP14[1,1]         :  -5.000                  ; Wear L3
```

The effect of setting data SD42930 on the `L3` component in the Y direction can be recognized without the additional complication caused by the frame rotation.

Example 9:

2 (tool length must be valued in the diameter axis with the factor 0.5) is the value of machine data:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).

X is diameter axis:

```
N10     def real _LEN[11]
N20     def real _CORVAL[3]
N30     $TC_DP1[1,1]= 500
N40     $TC_DP2[1,1]= 2
N50     $TC_DP3[1,1]= 3.
N60     $TC_DP4[1,1]= 4.
N70     $TC_DP5[1,1]= 5.
N80     _CORVAL[0] = 1.
N90     _CORVAL[1] = 1.
N100    _CORVAL[2] = 1.
N110    t1 d1 g18 g0 x0 y0 z0                    ; ==> MCS position X1.5 Y5 Z4
N120    r1 = settcor(_CORVAL, "g", 1, 1)
N130    t1 d1 x0 y0 z0                           ; ==> MCS position X2.5 Y6 Z5
N140    r3 = $TC_DP3[1,1]                        ; = 5. = (3.000 + 2. * 1.000)
N150    r4 = $TC_DP4[1,1]                        ; = 5. = (4.000 + 1.000)
N160    r5 = $TC_DP5[1.1]                        ; = 6. = (5.000 + 1.000)
N170    m30
```

The compensation of the tool length is to be 1 mm in each axis (N80 to N100).

1 mm is thus added to the original length in lengths L2 and L3.

Twice the compensation value (2 mm) is added to the original tool length in L1, in order to change the total length by 1 mm as required. If the positions approached in blocks N110 and N130 are compared, it can be seen that each axis position has changed by 1 mm.

17.16 Data lists

17.16.1 Machine data

17.16.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
18082	MM_NUM_TOOL	Number of tools that the NCK can manage (SRAM)
18088	MM_NUM_TOOL_CARRIER	Maximum number of the defined toolholders
18094	MM_NUM_CC_TDA_PARAM	Number of tool data (SRAM)
18096	MM_NUM_CC_TOA_PARAM	Number of data, per tool cutting edge for compile cycles (SRAM)
18100	MM_NUM_CUTTING_EDGES_IN_TOA	Tool offsets in the TOA area (SRAM)
18102	MM_TYPE_OF_CUTTING_EDGE	Type of D number programming (SRAM)
18105	MM_MAX_CUTTING_EDGE_NO	Maximum value of D number
18106	MM_MAX_CUTTING_EDGE_PERTOOL	Maximum number of D numbers per tool
18108	MM_NUM_SUMCORR	Number of all sum offsets in NCK
18110	MM_MAX_SUMCORR_PER_CUTTEDGE	Number of additive offsets per cutting edge
18112	MM_KIND_OF_SUMCORR	Properties of additive offsets in the TO area (SRAM)
18114	MM_ENABLE_TOOL_ORIENT	Assign orientation to cutting edges
18116	MM_NUM_TOOL_ENV	Tool environments in the TOA area (SRAM)

17.16.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20096	T_M_ADDRESS_EXT_IS_SPINO	Meaning of the address extension with T, M tool change
20110	RESET_MODE_MASK	Definition of initial control setting after RESET/part program end
20120	TOOL_RESET_VALUE	Tool length compensation at power-up (Reset/TP end)
20121	TOOL_PRESEL_RESET_VALUE	Preselected tool on RESET
20125	CUTMOD_ERR	Troubleshooting for the CUTMOD function
20126	TOOL_CARRIER_RESET_VALUE	Active toolholder on RESET
20127	CUTMOD_INIT	Initialize CUTMOD for POWER ON
20130	CUTTING_EDGE_RESET_VALUE	Tool cutting edge length compensation at power-up (Reset/TP end)
20132	SUMCORR_RESET_VALUE	Additive offset effective at RESET

Number	Identifier: $MC_	Description
20140	TRAFO_RESET_VALUE	Transformation data record at power up (Reset/TP end)
20180	TOCARR_ROT_ANGLE_INCR[i]	Rotary axis increment of the toolholder with orientation capability
20182	TOCARR_ROT_ANGLE_OFFSET[i]	Rotary axis offset of toolholder with orientation capability
20184	TOCARR_BASE_FRAME_NUMBER	Number of the basic frames to accept the table offset
20188	TOCARR_FINE_LIM_LIN	Limit linear fine offset TCARR
20190	TOCARR_FINE_LIM_ROT	Limit of the rotary fine offset TCARR
20202	WAB_MAXNUM_DUMMY_BLOCKS	Maximum number of blocks with no traversing motions with SAR
20204	WAB_CLEARANCE_TOLERANCE	Direction reversal for WAB
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Maximum value for DISC
20230	CUTCOM_CURVE_INSERT_LIMIT	Maximum value for intersection calculation with TRC
20240	CUTCOM_MAXNUM_CHECK_BLOCKS	Blocks for predictive contour calculation with tool radius compensation
20250	CUTCOM_MAXNUM_DUMMY_BLOCKS	Maximum number of blocks without traversing motion for TRC
20252	CUTCOM_MAXNUM_SUPPR_BLOCKS	Maximum number of blocks with compensation suppression
20256	CUTCOM_INTERS_POLY_ENABLE	Intersection process possible for polynomials
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20272	SUMCORR_DEFAULT	Initial setting of additive offset without program
20360	TOOL_PARAMETER_DEF_MASK	Definition of tool parameters
20390	TOOL_TEMP_COMP_ON	Activation of temperature compensation for tool length
20392	TOOL_TEMP_COMP_LIMIT	Maximum temperature compensation for tool length
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements
21080	CUTCOM_PARALLEL_ORI_LIMIT	Minimum angle (path tangent and tool orientation) for 3D tool radius compensation
22530	TOCARR_CHANGE_M_CODE	M code for change of toolholder
22550	TOOL_CHANGE_MODE	New tool compensations with M function
22560	TOOL_CHANGE_M_CODE	M function for tool change
22562	TOOL_CHANGE_ERROR_MODE	Response when errors occur at tool change
24558	TRAFO5_JOINT_OFFSET_PART_1	Vector of kinematic offset in table, transformation 1
24658	TRAFO5_JOINT_OFFSET_PART_2	Vector of kinematic offset in table, transformation 2
28085	MM_LINK_TOA_UNIT	Assigning ´the TO unit to a channel (SRAM)

17.16.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32750	TEMP_COMP_TYPE	Temperature compensation type

17.16.2 Setting data

17.16.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42442	TOOL_OFFSET_INCR_PROG	Tool length compensation
42470	CRIT_SPLINE_ANGLE	Core limit angle, for compressor
42480	STOP_CUTCOM_STOPRE	Alarm response for tool radius compensation and preprocessing stop
42494	CUTCOM_ACT_DEACT_CTRL	Approach and retraction behavior for tool radius compensation
42496	CUTCOM_CLSDT_CONT	Behavior of the tool radius compensation for closed contour
42900	MIRROR_TOOL_LENGTH	Sign change, tool lengths when mirroring
42910	MIRROR_TOOL_WEAR	Sign change, tool wear when mirroring
42920	WEAR_SIGN_CUTPOS	Sign of wear for tools with cutting edge position
42930	WEAR_SIGN	Sign of the wear
42935	WEAR_TRANSFORM	Transformations for tool components
42940	TOOL_LENGTH_CONST	Change of tool length components for change of plane
42950	TOOL_LENGTH_TYPE	Assignment of the tool length offset independent of tool type
42960	TOOL_TEMP_COMP	Temperature compensation value in relation to tool
42974	TCARR_FINE_CORRECTION	Fine offset `TCARR` on/off
42984	CUTDIRMOD	Modification of $P_AD[2] or $P_AD[11]

17.16.3 Signals

17.16.3.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
T function 1 change	DB21, ...DBX61.0	-
D function 1 change	DB21, ...DBX62.0	-
T function 1	DB21, ...DBB116-119	DB2500.DBD2000
D function 1	DB21, ...DBB128-129	DB2500.DBD5000
Active G function of group 7	DB21, ...DBB214	DB3500.DBB6
Active G function of group 16	DB21, ...DBB223	DB3500.DBB15
Active G function of group 17	DB21, ...DBB224	DB3500.DBB16
Active G function of group 18	DB21, ...DBB225	DB3500.DBB17
Active G function of group 23	DB21, ...DBB230	DB3500.DBB22

Z1: NC/PLC interface signals 18

18.1 Various interface signals and functions (A2)

18.1.1 Signals from PLC to NC (DB10)

DB10 DBX56.4 - DBX56.7	Key-operated switch positions 0 to 3
Edge evaluation: No	Signal(s) updated: Cyclic
Significance of signal	Depending on the key-operated switch position, access to certain elements in the NCK can be enabled or disabled. • Key-operated switch position 0 represents the lowest access rights • Key-operated switch position 3 represents the highest access rights The NC/PLC interface signals of key-operated switch positions 1 to 3 can either be directly specified from the key-operated switch on the machine control panel or from the PLC user program. It is only permissible to set one bit. If several bits are set simultaneously, the control internally activates switch position 3.

Key-operated switch position	DBX56.7	DBX56.6	DBX56.5	DBX56.4
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

Corresponding to ...	Machine data for protection levels: MD11612, MD51044 - MD51064, MD51070 - MD51073, MD51199 - MD51211, MD51215 - MD51225, MD51235 Disabling using a password

18.1.2 Selection/Status signals from HMI to PLC (DB10)

DB10 DBX103.0	Remote diagnosis active
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Remote diagnosis (optional) is active, i.e. the control is operated via an external PC.
Signal state 0 or edge change 1 → 0	Remote diagnosis is not active.

DB10 DBX103.5	AT box ready
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The AT box for expansion modules is ready.
Signal state 0 or edge change 1 → 0	The AT box is not ready. An expansion module conforming to the AT specification has either no functionality or restricted functionality.

DB10 DBX103.6	HMI temperature limit
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The environment conditions of the limit value are in the permitted tolerance range of 5 to 55° C.
Signal state 0 or edge change 1 → 0	The temperature range was either exceeded, or the temperature fell below the limit. The temperature monitor has responded and the PCU is disabled.

DB10 DBX103.7	HMI battery alarm
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The battery monitor has responded. Power failure can cause the loss of recently changed data and a correct device configuration. An appropriate alarm is issued. The buffer battery should be checked. An insufficient battery voltage also affects the current time on the user interface.
Signal state 0 or edge change 1 → 0	No HMI battery alarm is present.
Additional references	Operator Components Manual (PCU)

18.1.3 Signals from the NC to the PLC (DB10)

DB10 DBX104.7	NCK CPU ready
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The NCK CPU is ready and registers itself cyclically with the PLC. After a correct initial start and the first complete OB1 cycle (initial setting cycle) the PLC and NCK continuously exchange sign-of-life signals. The PLC basic program sets the interface signal "NCK CPU Ready" to 1.

DB10 DBX104.7	**NCK CPU ready**
Signal state 0	The NCK CPU is not ready. If a sign-of-life is not received from the NCK, the PLC/NCK interface is neutralized by the PLC basic program and the interface signal "NCK CPU Ready" is set to 0. The following measures are introduced by the PLC basic program: • Status signals from NCK to PLC (user interface) are deleted (cleared) • Change signals for help functions are deleted • Cyclic processing of the user interface PLC to NCK is terminated.
Additional references	• Diagnostics Manual • Function Manual Basic Functions; Basic PLC Program

DB10 DBX108.3	**SINUMERIK Operate at OPI ready**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	SINUMERIK Operate is ready and cyclically logs on with the NCK.
Signal state 0	SINUMERIK Operate is not ready.
Additional references	Diagnostics Manual

DB10 DBX108.5	**Drives in cyclic operation**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	For all machine axes of the NC, the corresponding drives are in the cyclic operation, i.e. they cyclically exchange PROFIdrive telegrams with the NC.
Signal state 0	For at least one machine axis of the NC, the corresponding drive is not in cyclic operation, i.e. it is not cyclically exchanging PROFIdrive telegrams with the NC.

DB10 DBX108.6	**Drive ready**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	For **all** machine axes of the NC, the corresponding drives are ready: DB31, ... DBX93.5 == 1 (DRIVE ready)
Signal state 0	For at least one machine axis of the NC, the corresponding drive is **not** ready: DB31, ... DBX93.5 == 0 (DRIVE ready)
Corresponding to ...	DB31, ... DBX93.5 (DRIVE ready)

DB10 DBX108.7	**NC ready**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The control system is ready. This interface signal is an image of the relay contact "NC Ready". This signal is set if: • Relay contact "NC Ready" is closed • All the voltages in the controller have been established • The controller is in the cyclic mode
Signal state 0	The controller is not ready. The relay contact "NC Ready" is open. The following faults will cause NC Ready to be canceled: • Undervoltage and overvoltage monitoring function has responded • Individual components are not ready (NCK CPU Ready) • NCK CPU watchdog If the signal "NC Ready" goes to 0 the following measures are introduced by the controller if they are still possible: • The controller enable signals are withdrawn (this stops the drives) • The following measures are introduced by the PLC basic program: – Status signals from NCK to PLC (user interface) are deleted (cleared) – Change signals for help functions are deleted – Cyclic processing of the PLC to NCK user interface is terminated For further information see References! The controller is not ready again until after POWER ON.
Corresponding to ...	Relay contact "NC Ready"
Additional references	• Diagnostics Manual • Function Manual Basic Functions; Basic PLC Program

DB10 DBX109.0	**NCK alarm is active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	At least one NCK alarm is present. This is a group signal for the interface signals of all existing channels: DB21, ... DBX36.6 (channel-specific NCK alarm is active)
Signal state 0	No NCK alarm is active.
Corresponding to ...	DB21, ... DBX36.6 (channel-specific NCK alarm is active) DB21, ... DBX36.7 (NCK alarm with processing stop present)
Additional references	Diagnostics Manual

DB10 DBX109.5	NCU heat sink temperature alarm
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	In the NCU, the limit values of the heat sink temperature was exceeded. The heat sink temperature has been activated and a steady operation of the operator panel is no longer guaranteed.
Signal state 0	The heat sink monitoring function of the NCU has not responded.

DB10 DBX109.6	Air temperature alarm
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The ambient temperature or fan monitoring function has responded. This may be due to the following causes: • The temperature monitoring has identified an ambient temperature that is too high (approx. 60 °C). Alarm 2110 "NCK temperature alarm" is output. • The speed monitoring of the 24 VDC fan used to cool the module has responded. Alarm 2120 "NCK fan alarm" is output. Measures: Replace the fan and/or ensure that additional ventilation is provided. When a temperature or fan error responds, a relay contact (terminal 5.1, 5.2 or 5.1, 5.3) in the infeed/regenerative feedback unit is activated which can be evaluated by the customer.
Signal state 0	Neither the temperature monitoring nor the fan monitoring has responded.
Application example(s)	Appropriate measures can be initiated by the PLC user program if the temperature or fan monitoring is activated.
Corresponding to ...	When a temperature or fan error responds, a relay contact (terminal 5.1, 5.2 or 5.1, 5.3) in the infeed/regenerative feedback unit is activated. This can be evaluated.
Additional references	Diagnostics Manual

DB10 DBX109.7	**NCK battery alarm**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The NCK battery voltage monitoring function has responded. This may be due to the following causes: • The battery voltage is within the pre-warning limit range (approx. 2.7 to 2.9 V). Alarm 2100 "NCK battery warning threshold reached" has been triggered. Refer to References for effects and measure. • The battery voltage is below the pre-warning limit range (≤ 2.6 V). Alarm 2101 "NCK battery alarm" is signaled in cyclic operation. Effects: A supply voltage failure - e.g. when the controller is switched off - would result in the loss of battery-buffered data (e.g. part program memory, variables, machine data ...). Measure: Refer to additional References. • When the controller powered-up, it was identified that the battery voltage was below the pre-warning limit range (≤ 2.6 V). Alarm 2102 "NCK battery alarm" is output; NC ready and mode group ready are not issued. Effects: Some of the battery-buffered data may already have been lost! Measure: Refer to additional References.
Signal state 0	The battery voltage is above the lower limit value (normal condition).
Special cases, errors, ...	The NCK batteries should only be replaced while the NC is switched on to avoid data loss because of no memory backup.
Additional references	• Diagnostics Manual • Equipment Manual NCU

18.1.4 Signals to Operator Panel (DB19)

DB19 DBX0.0	**Screen bright**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The screen blanking is disabled.
Signal state 0 or edge change 1 → 0	The screen blanking remains in effect.
Corresponding to ...	DB19 DBX0.1 (darken screen)

DB19 DBX0.1	**Darken screen**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The screen is darkened by the PLC user program. The automatic screen brightening/darkening is therefore ineffective: i.e. the screen does not brighten up automatically on actuating the keyboard.
Signal state 0 or edge change 1 → 0	The screen is controlled by the PLC user program "bright". In this signal state, the screen brightening/darkening setting can be derived by the control via the keyboard automatically. The screen is darkened if no key is pressed for a period defined via the following machine data on the keyboard: MD9006 $MM_DISPLAY_BLACK_TIME (time to darken the screen) The screen is brightened the next time a key on the operator panel front is pressed.
Application example(s)	Screen saver
Special cases, errors, ...	Notice: The keyboard of the operator panel front continues to be effective if the interface signal: DB19 DBX0.1 (darken screen) = 1 We therefore recommend that this is disabled using the interface signal: DB19 DBX0.2 (key disable)
Corresponding to ...	DB19 DBX0.2 (key disable) MD9006 $MM_DISPLAY_BLACK_TIME (time to darken the screen)

DB19 DBX0.2	**Key disable**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The keyboard is disabled for the user.
Signal state 0 or edge change 1 → 0	The keyboard is enabled for the user.
Application example(s)	If the screen is darkened with the interface signal: DB19 DBX0.1 (darken screen), the keyboard should be actuated simultaneously with the interface signal: DB19 DBX0.2 (key disable) to avoid an unintended operation.
Corresponding to ...	DB19 DBX0.1 (darken screen)

DB19 DBX0.3	**Clear cancel alarms**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Clear error key on the machine control panel is pressed. All cancel alarms of the NCKs and the control panel are then acknowledged. The PLC application acknowledges the PLC alarms itself. Power On and Reset alarms remain active on the NCK until the cause of the error has been removed.
Signal state 0 or edge change 1 → 0	Clear error key on the machine control panel is not pressed.
Functionality	Only valid for HMI Advanced.
Corresponding to ...	DB19 DBX20.3 (cancel alarm cleared)

DB19 DBX0.4	**Clear recall alarms**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Clear error key on the machine control panel is pressed. All cancel alarms of the NCKs and the control panel are then acknowledged. The PLC application acknowledges the PLC alarms itself. POWER On and Reset alarms remain active on the NCK until the cause of the error has been removed.
Signal state 0 or edge change 1 → 0	Clear error key on the machine control panel is not pressed.
Application example(s)	Applies to HMI Advanced only
Corresponding to ...	DB19 DBX20.4 (recall alarm cleared)

DB19 DBX0.7	**Actual values in WCS, 0 = MCS**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The PLC selects the display of actual values in the workpiece coordinate system (WCS). This means that when the machine area is selected, the WCS display is activated; i.e. the machine and the supplementary axes as well as their actual positions and distances-to-go are displayed in the WCS in the "Position" window. The interface signal is only evaluated when it enters the basic machine screen; this means that the operator, within the machine area, can toggle as required between the particular coordinate systems using the softkeys "actual values MCS" and "actual values WCS".
Signal state 0 or edge change 1 → 0	This means that when the machine area is selected the coordinate system previously selected (WCS or MCS) is reactivated and displayed.
Application example(s)	Using the interface signal:DB19, DBX0.7 (actual value in WCS) = 1 each time that the machine area is re-selected, the workpiece coordinate system display frequently required by the operator (WCS), is selected.
Corresponding to ...	DB19 DBX20.7 (changeover MCS/WCS)
Additional references	Operation Guide HMI (corresponding to the used software)

DB19 DBB6	Analog spindle 1, utilization in percent
Edge evaluation: No	Signal(s) updated: Cyclically

DB19 DBB7	Analog spindle 2, utilization in percent
Edge evaluation: No	Signal(s) updated: Cyclically

DB19 DBB8	Channel number of the machine control panel to HMI
Edge evaluation: No	Signal(s) updated: Cyclically

DB19 DBB10	PLC hard keys (value range 1 ...255, 0 is the initial state)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Program area selection active
Signal state 0 or edge change 1 → 0	Program area selection inactive

DB19 DBX13.5	Unload part program
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Unload active
Signal state 0 or edge change 1 → 0	Unload inactive
Application example(s)	A file transfer can be initiated using the hard disk.

DB19 DBX13.6	**Load part program**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Load active
Signal state 0 or edge change 1 → 0	Load inactive
Application example(s)	A file transfer can be initiated using the hard disk.

DB19 DBX13.7	**Part program selection**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Selection active
Signal state 0 or edge change 1 → 0	Selection inactive
Application example(s)	A file transfer can be initiated using the hard disk.

DB19 DBX14.0 - DBX14.6	**PLC index**
Edge evaluation: No	Signal(s) updated: Cyclically
Description	This byte for control of the RS-232-C describes the PLC index for the standard control file that specifies the axis, channel or TO number.
Application example(s)	Dependent on: DB19 DBX14.7=0 → Act. FS: PLC index that specifies axis, channel or TO No. DB19 DBX14.7=1 → Pas. FS: PLC index for the user control file

DB19 DBX14.7	**Active or passive file system**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Passive file system
Signal state 0 or edge change 1 → 0	Active file system

DB19 DBB15	PLC line offset
Edge evaluation: No	Signal(s) updated: Cyclically
Description	This byte to control the RS-232-C interface defines the line of the standard or user control file in which the control file to be transferred is specified.
Application example(s)	Dependent on: DB19 DBX14.7=0 → Act. FS: PLC line offset in a standard control file DB19 DBX14.7=1 → Pas. FS: PLC line offset in a user control file

DB19 DBX16.0 - DBX16.6	PLC index for the user control file
Edge evaluation: No	Signal(s) updated: Cyclically
Description	This byte for controlling file transfer via hard disk defines the index for the control file (job list).
Application example(s)	Dependent on: DB19 DBX14.7=0 → Act. FS: PLC index for the standard control file DB19 DBX14.7=1 → Pas. FS: PLC index for the user control file

DB19 DBX16.7	Active or passive file system
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Passive file system
Signal state 0 or edge change 1 → 0	Active file system
Application example(s)	with HMI Advanced always 1

DB19 DBB17	PLC line offset in the user control file
Edge evaluation: No	Signal(s) updated: Cyclically
Description	This byte for controlling file transfer via the hard disk defines the line of the user control file in which the control file to be transferred is located.
Application example(s)	Dependent on: DB19 DBX14.7=0 → Act. FS: PLC line offset in a standard control file DB19 DBX14.7=1 → Pas. FS: PLC line offset in a user control file

DB19 DBX44.0	Mode change disable
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Mode change disable active
Signal state 0 or edge change 1 → 0	Mode change disable active

DB19 DBX45.0	FC9 Out: Active
Edge evaluation: No	Signal(s) updated: Cyclically

DB19 DBX45.1	FC9 Out: Done
Edge evaluation: No	Signal(s) updated: Cyclically

DB19 DBX45.2	FC9 Out: Error
Edge evaluation: No	Signal(s) updated: Cyclically

DB19 DBX45.3	FC9 Out: StartError
Edge evaluation: No	Signal(s) updated: Cyclically

18.1.5 Signals from operator control panel (DB19)

DB19 DBX20.1	Screen dark
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The screen is darkened.
Signal state 0	The screen is not darkened.
Application example(s)	Using this IS, the PLC can identify whether the screen was switched dark using the interface signal: DB19, DBX0.1 (darken screen) or using the machine data: MD9006 $MM_DISPLAY_BLACK_TIME (screen blanking time) .
Corresponding to ...	MD9006 $MM_DISPLAY_BLACK_TIME (screen blanking time)

DB19 DBX20.3	Cancel alarm deleted
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Cancel alarm deleted active
Signal state 0	Cancel alarm deleted inactive Note: The signal is not reset automatically, it must be set by the user via the PLC user program.
Application example(s)	Applies only to HMI Advanced

DB19 DBX20.4	Recall alarm deleted
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Recall alarm deleted inactive
Signal state 0	Recall alarm deleted inactive Note: The signal is not reset automatically, it must be set by the user via the PLC user program.
Application example(s)	Applies only to HMI Advanced

DB19 DBX20.6	**Simulation active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	On entry to simulation = 1
Signal state 0	On exit from simulation = 0
Application example(s)	Can be evaluated by machine manufacturer in order to activate the test on NC start.

DB19 DBX20.7	**Switch over MCS/WCS**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The coordinate system is switched over from workpiece coordinate system (WCS) to machine coordinate system (MCS) or from MCS to WCS. After it has been set, the signal is active for one PLC cycle.
Signal state 0	No effect
Application example(s)	The interface signal: DB19, DBX20.7 (switch over MCS/WCS) must be transferred to the interface signal: DB19, DBX0.7 (actual value in WCS) in order that the switchover becomes effective.
Corresponding to ...	DB19, DBX0.7 (actual value in WCS)

DB19 DBB22	**Displayed channel number from HMI**
Edge evaluation: No	Signal(s) updated: Cyclically

DB19 DBB24	**Actual image number of the JobShop interface**
Edge evaluation: No	Signal(s) updated: Cyclically

DB19 DBX26.1	**OK Part program handling status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Transfer correctly completed
Signal state 0	Transfer completed with error
Additional references	Commissioning Manual, SINUMERIK Operate (IM9), Section "PLC functions" > "Program selection"
Corresponding to ...	DB19.DBB13 (program selection: Requirement) DB19.DBB16 (program selection: Control file index) DB19.DBB17 (program selection: Program list index) DB19.DBB27 (program selection: Error handling)

DB19 DBX26.2	**Error (part program handling status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Transfer completed with error
Signal state 0	Transfer correctly completed
Additional references	Commissioning Manual, SINUMERIK Operate (IM9), Section "PLC functions" > "Program selection"
Corresponding to ...	DB19.DBB13 (program selection: Requirement) DB19.DBB16 (program selection: Control file index) DB19.DBB17 (program selection: Program list index) DB19.DBB27 (program selection: Error handling)

DB19 DBX26.3	**Active (part program handling status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Job in progress
Signal state 0	No job in progress
Additional references	Commissioning Manual, SINUMERIK Operate (IM9), Section "PLC functions" > "Program selection"
Corresponding to ...	DB19.DBB13 (program selection: Requirement) DB19.DBB16 (program selection: Control file index) DB19.DBB17 (program selection: Program list index) DB19.DBB27 (program selection: Error handling)

DB19 DBX26.5	**Unload (part program handling status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Unload active
Signal state 0	Unload inactive
Additional references	Commissioning Manual, SINUMERIK Operate (IM9), Section "PLC functions" > "Program selection"
Corresponding to ...	DB19.DBB13 (program selection: Requirement) DB19.DBB16 (program selection: Control file index) DB19.DBB17 (program selection: Program list index) DB19.DBB27 (program selection: Error handling)

DB19 DBX26.6	**Load (part program handling status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Load active
Signal state 0	Load inactive
Additional references	Commissioning Manual, SINUMERIK Operate (IM9), Section "PLC functions" > "Program selection"
Corresponding to ...	DB19.DBB13 (program selection: Requirement) DB19.DBB16 (program selection: Control file index) DB19.DBB17 (program selection: Program list index) DB19.DBB27 (program selection: Error handling)

DB19 DBX26.7	**Select (part program handling status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Selection active
Signal state 0	Selection inactive
Additional references	Commissioning Manual, SINUMERIK Operate (IM9), Section "PLC functions" > "Program selection"
Corresponding to ...	DB19.DBB13 (program selection: Requirement) DB19.DBB16 (program selection: Control file index) DB19.DBB17 (program selection: Program list index) DB19.DBB27 (program selection: Error handling)

DB19 **DBB27**	**Error program handling**	
Edge evaluation: No		Signal(s) updated: Cyclically
Description	Error codes: aaa	
	Value	**Meaning**
	0	No error
	1	Invalid number for the control file (value in DB19.DBB16 < 127 or invalid)
	3	Control file "plc_proglist_main.ppl" not found (value in DB19.DBB16 invalid)
	4	Invalid index in control file (incorrect value in DB19.DBB17)
	5	Job list in the selected workpiece could not be opened
	6	Error in job list. (Job list Interpreter returns error)
	7	Job list Interpreter returns empty job list
Additional references	Commissioning Manual, SINUMERIK Operate (IM9), Section "PLC functions" > "Program selection"	
Corresponding to ...	DB19.DBB13 (program selection: Requirement) DB19.DBB16 (program selection: Control file index) DB19.DBB17 (program selection: Program list index) DB19.DBB26 (program selection: Acknowledgement)	

18.1.6 Signals to channel (DB21, ...)

DB21, ... DBX6.2	Delete distance-to-go (channel-specific)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Delete distance-to-go (channel-specific): IS "Delete distance-to-go (channel-specific)" for path axes is only active in AUTOMATIC mode. The rising edge of the interface signal is only effective for the axes involved in the geometry grouping. These are also stopped with a ramp stop and their distance-to-go deleted (setpoint - actual value difference). Any remaining following error is still corrected. The next program block is then started. IS "Delete distance-to-go (channel-specific)" is therefore ignored by positioning axes. Remark: IS "Delete distance-to-go" does not influence the running dwell time in a program block with dwell time.
Signal state 0 or edge change 1 → 0	No effect
Signal irrelevant for ...	Positioning axes
Application example(s)	To terminate motion because of an external signal (e.g. measuring probe)
Special cases, errors, ...	Delete distance-to-go (channel-specific) When the axes have been stopped with IS "Delete distance-to-go" the next program block is prepared with the new positions. After a "Delete distance-to-go", geometry axes thus follow a different contour to the one originally defined in the part program. If G90 is programmed in the block after "Delete distance-to-go" it is at least possible to approach the programmed absolute position. On the other hand, with G91, the position originally defined in the part program is no longer reached in the following block.
Corresponding to ...	DB31, ... DBX2.2 (delete distance-to-go (axis-specific))

18.1.7 Signals from channel (DB21, ...)

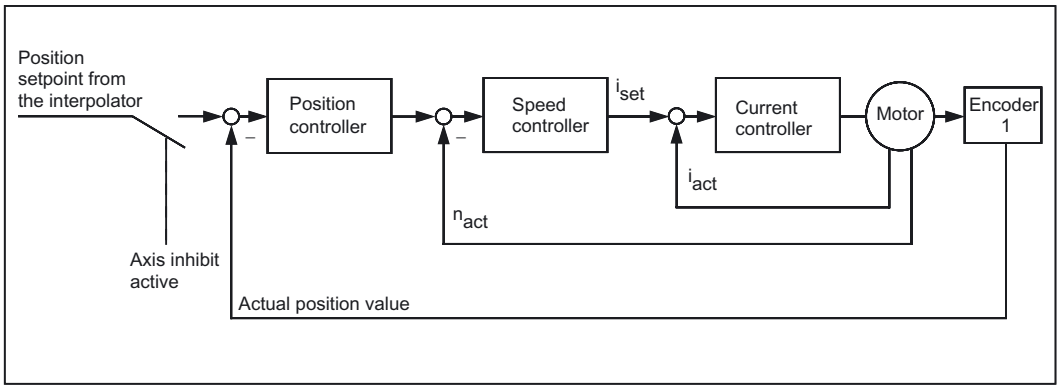
DB21, ... DBX36.6	**Channel-specific NCK alarm is active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	At least one NCK alarm is present for this channel. Thus the following group interface signal is also set: DB10 DBX109.0 (NCK alarm is present) The PLC user program can interrogate whether processing for the channel in question has been interrupted because of an NCK channel: DB21, ... DBX36.7 (NCK alarm with processing stop present).
Signal state 0 or edge change 1 → 0	No NCK alarm is active for this channel.
Corresponding to ...	DB21, ... DBX36.7 (NCK alarm with processing stop pending) DB10 DBX109.0 (NCK alarm pending)
Additional references	Diagnostics guide

DB21, ... DBX36.7	**NCK alarm with processing stop present**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	At least one NCK alarm - which is causing a processing stop of the part program running in this channel - is active.
Signal state 0 or edge change 1 → 0	There is no alarm active in this channel that is causing a processing stop.
Application example(s)	With this alarm a program interruption because of an NCK alarm can be recognized immediately by the PLC user program and the necessary steps introduced.
Corresponding to ...	DB21, ... DBX36.6 (channel-specific NCK alarm pending)
Additional references	Diagnostics guide

18.1.8 Signals to axis/spindle (DB31, ...)

DB31, ... DBX1.0	Drive test travel enable
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Traversing motions of the axis are enabled for the drive test. Feedback signal of the safety handshake at the start of the NC function generator. The NC has requested the traversing enable for the axis at the start of the function generator with the following interface signal: DB31, ... DBX61.0 = 1 (drive test travel request) The PLC user program sets the current interface signal as feedback signal to the NC that the axis can be traversed: DB31, ... DBX1.0 = 1 (drive test travel enable) Only the PLC can decide on the traversing enable of an axis.
Signal state 0	Traversing motions of the axis are disabled for the drive test.
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX1.3	Axis/spindle disable
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	**Axis** No setpoints are output to the position controller. The traversing motion of the axis is disabled. However, the axis is still in closed-loop position control and any remaining following error is compensated. If the axis is traversed, the position setpoint and velocity setpoint are displayed as actual position and actual velocity on the user interface. No setpoints are transferred to the drive, the machine axis therefore does not traverse. With RESET, the display of the actual value is set to the actual value of the machine axis. Travel commands are output to the NC/PLC interface. If the interface signal is set for a traversing axis, this is stopped in compliance with the acceleration characteristic and an alarm is displayed. **Spindle** No speed setpoints are output to the speed controller in open-loop control mode. No position partial setpoints are output to the position controller in positioning mode. The traversing motion of the spindle is disabled. The speed setpoint is displayed as the actual speed value on the user interface. If the interface signal is set for a traversing spindle, this is stopped in compliance with the acceleration characteristic and an alarm is displayed.

DB31, ... DBX1.3	**Axis/spindle disable**
Signal state 0	The reset of the interface signal does not take effect until the axis/spindle is stationary, i.e. an interpolation setpoint is no longer present. **Axis** Position setpoints are transferred to the position controller cyclically. If the interface signal was set, traversing can be performed normally immediately after the reset of the axis. **Spindle** Speed setpoints are transferred to the speed controller cyclically. If the interface signal was set, the effective spindle disable can only be canceled after the reset through RESET or M2 and a program restart.
	

<table>
<tr><td>DB31, ...
DBX1.3</td><td>Axis/spindle disable</td></tr>
<tr><td>Application example(s)</td><td>Prevention of traversing motions of the axis or spindle when running-in and testing a new part program.</td></tr>
<tr><td>Special cases, errors, ...</td><td>If the interface signal is set, the following interface signals have no effect with regard to the braking of the axis/spindle:
• DB31, ... DBX2.1 (controller enable)
• DB2 ... (feed/spindle stop)
• DB31, ... DBX12.0-12.1 (hardware limit switch)
The axis/spindle can still be switched to the "hold" or "follow up" state
(see DB31, ... DBX1.4 (follow-up mode)).
Notes
• The output of setpoints on the drive is disabled with the interface signal.
• A brief pulse can bring a traversing axis to a standstill. The axis will not move again in this block, but only when the next block is reached.
• Re-synchronization takes place automatically on the next traversing command for this axis, i.e. the axis traverses the remaining distance-to-go.
Example:
<pre>N10 G0 X0 Y0
N20 G1 F1000 X100
N30 Y100
N40 X200</pre>Regarding N20: At position X20, the interface signal "Axis disable" is briefly set. The X axis is braked to standstill.
Regarding N40: The X axis traverses from the last position (approx. 20 mm) to the programmed position 200 mm.
Effects of the interface signal for spindle or axis couplings:
<table>
<tr><th>LS/LA[1]</th><th>FS/FA[1]</th><th>Coupl.[2]</th><th>Effect</th></tr>
<tr><td>0</td><td>0</td><td>Off</td><td>Setpoints of axes are output</td></tr>
<tr><td>0</td><td>1</td><td>Off</td><td>No setpoint output for FS/FA</td></tr>
<tr><td>1</td><td>0</td><td>Off</td><td>No setpoint output for LS/LA</td></tr>
<tr><td>1</td><td>1</td><td>Off</td><td>No setpoint output for FS/FA and LS/LA</td></tr>
<tr><td>0</td><td>0</td><td>On</td><td>Setpoints of axes are output</td></tr>
<tr><td>0</td><td>1</td><td>On</td><td>Disable not effective for FS/FA</td></tr>
<tr><td>1</td><td>0</td><td>On</td><td>Disable also effective for FS/FA</td></tr>
<tr><td>1</td><td>1</td><td>On</td><td>No setpoint output for FS/FA and LS/LA</td></tr>
<tr><td colspan="4">1) Status of the interface signal; LS/LA: Leading spindle/axis; FS/FA: Following spindle/axis
2) Status of the coupling between leading and following spindle/axis</td></tr>
</table></td></tr>
<tr><td>Corresponding to ...</td><td>DB21, ... DBX33.7 (program test active)</td></tr>
<tr><td>Additional references</td><td>Behavior in synchronous mode:
Function Manual, Extended Functions; Synchronous Spindle (S3)</td></tr>
</table>

DB31, ... DBX1.4	**Follow-up mode**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Follow-up mode is active. • The position setpoint is continuously tracked: Position setpoint = actual position value. • Feedback signal for the active follow-up mode: DB31, ... DBX61.3 = 1 (follow-up active) • Zero speed and clamping monitoring are not active. • When the closed-loop control system is reactivated, a control-internal repositioning operation is performed (REPOSA: linear approach with all axes) to the last programmed position if a part program is active.
Signal state 0	Follow-up mode is not active. When "controller enable" is removed the previous position setpoint is kept in the control. If the axis/spindle is moved out of position during this time a following error occurs between the position setpoint and the actual position value. This position difference is reduced to zero immediately by issuing "controller enable" so that the previous setpoint position is restored. Then, all the other axis motions start from the setpoint position valid before "controller enable" was removed. When the position control is switched on again the axis may make a speed setpoint jump. Zero speed and clamping monitoring are active. To switch off the zero speed monitoring, the following interface signal should be set when clamping an axis: DB31, ... DBX2.3 = 1 (clamping operation running). In the "hold" state, the interface signal: DB31, ... DBX61.3 (follow-up active) is set to a 0 signal.
Special cases, errors, ...	If the controller enable is canceled in the control because of faults, the "hold" state should be activated for the axis before the NC start after the queued alarms have been successfully deleted and the accompanying the controller enable set: DB31, ... DBX1.4 = 0 (follow-up mode). Otherwise, for an NC start and active follow-up mode, the traversing distance of the previous NC block would not be executed due to the internal delete distance-to-go. **Notice:** During the transition from the "follow-up" state to the "hold" state or when the controller enable is set in position control, delete distance-to-go is activated in the control. As a consequence, for example, a block in which only this axis is traversed is terminated directly.
Corresponding to ...	DB31, ... DBX2.1 (controller enable) DB31, ... DBX2.3 (clamping in progress) DB31, ... DBX61.3 (follow-up active)

<table>
<tr><td>DB31, ...
DBX1.5 - DBX1.6</td><td colspan="3">Position measuring system 1 (PMS1) / Position measuring system 2 (PMS2)</td></tr>
<tr><td>Edge evaluation: No</td><td colspan="3">Signal(s) updated: Cyclically</td></tr>
<tr><td>Signal state 1</td><td colspan="3">The position measuring system is enabled</td></tr>
<tr><td>Signal state 0</td><td colspan="3">The position measuring system is disabled</td></tr>
<tr><td rowspan="6">Signal state overview</td><td>PMS1</td><td>PMS2</td><td>Effect</td></tr>
<tr><td>1</td><td>0</td><td>Position measuring system 1 is active:
• Position measuring system 1 is used for position control.
• If position measuring system 2 is also available (MD30200 $MA_NUM_ENCS == 2), its actual position value is also acquired.</td></tr>
<tr><td>0</td><td>1</td><td>Position measuring system 2 is active:
• Position measuring system 2 is used for position control.
• If position measuring system 1 is also available (MD30200 $MA_NUM_ENCS == 2), its actual position value is also acquired.</td></tr>
<tr><td>1</td><td>1</td><td>• Position measuring system 1 is used for position control.
• If position measuring system 2 is also available (MD30200 $MA_NUM_ENCS == 2), its actual position value is also acquired.</td></tr>
<tr><td>0</td><td>0</td><td>Position measuring systems 1 and 2 are inactive:
• There is no actual value acquisition.
• The monitoring of the position measuring system has been deactivated.
• The following interface signals are reset:
– DB31, ... DBX60.4/5 == 0 (referenced/synchronized)
– DB31, ... DBX61.5 (position controller active)
– DB31, ... DBX61.6 (speed controller active)
– DB31, ... DBX61.7 (current controller active)
After parking has been completed, the axis must be referenced again.</td></tr>
<tr><td colspan="3">Notes
• If the interface signal of the active position measuring system is reset for a traversing axis, the axis is stopped with a ramp stop without the controller enable being canceled internally.
• If a speed-controlled spindle does not have a position measuring system, the "Controller enable" interface signal must be set:
DB31, ... DBX2.1 == 1 (controller enable)</td></tr>
</table>

DB31, ... DBX1.5 - DBX1.6	**Position measuring system 1 (PMS1) / Position measuring system 2 (PMS2)**
Application example(s)	1. Switching over from position measuring system 1 to positioning measuring system 2 (and vice versa). If the axis was referenced in both position measuring systems and the limit frequency of the used measured value encoder has not been exceeded in the meantime, i.e. DB31, ... DBX60.4 and 60.5 == 1 (referenced/synchronized 1 and 2) , a new reference point approach is not required after the switchover. On switchover, the current difference between position measuring system 1 and 2 is traversed immediately. A tolerance band in which the deviation between the two actual values may lie at the switchover can be specified with the following machine data: MD36500 $MA_ENC_CHANGE_TOL (maximum tolerance for position actual value switchover) If the actual value difference is greater than the tolerance, there is no switchover and alarm 25100 "Measuring system switchover not possible" is displayed. 2. Parking axis (i.e. no position measuring system is active): The position measuring system monitoring is switched off when the measured value encoder is removed. 3. Switch off position measuring system: When position measuring system 1 or 2 is switched off, the associated interface signal: DB31, ... DBX60.4/60.5 (referenced/synchronized 1/2) is reset. 4. Reference point approach: Reference point approach of the axis is executed with the selected position measuring system. Every position measuring system must be separately referenced.
Special cases, errors, ...	If the state "parking axis" is active, the following interface signal is ignored at NC start for this axis: DB31, ... DBX60.4/60.5 (referenced/synchronized 1/2).
Corresponding to ...	DB31, ... DBX60.4/.5 (referenced/synchronized 1/2) DB31, ... DBX61.6 (speed controller active) DB31, ... DBX2.1 (controller enable) MD36500 $MA_ENC_CHANGE_TOL (max. tolerance on actual position value switchover) MD30200 $MA_NUM_ENCS (number of encoders)
Additional references	Function Manual, Basic Functions; Velocities, Setpoint / Actual Value Systems, Closed-Loop Control (G2)

DB31, ... DBX2.1	**Controller enable**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Controller is enabled. The position control loop is closed and the axis/spindle is in closed-loop control. Feedback: DB31, ... DBX61.5 = 1 (position controller active) If the axis/spindle was referenced before resetting the interface signal, the axis/spindle does not have to be re-referenced after the interface signal is set again. Supplementary condition: The limit frequency of the active measuring system must not be exceeded in the meantime. **Note** If the axis/spindle was moved from its position during the time in which the controller enable was not set, the behavior when the controller enable is set depends on the interface signal "follow-up mode": • DB31, ... DBX1.4 == 1 (follow-up mode) The position control is implemented on the current position • DB31, ... DBX1.4 == 0 (follow-up mode) The position control is implemented on the last position before the reset the controller enable
Signal state 0	Controller is not enabled. The behavior when the "controller enable" is removed depends on whether the axis/spindle is stationary or traversing at this time: • Axis/spindle stationary: – The position control loop of axis is opened. – For DB31, ... DBX1.4 == 1 (follow-up mode) ⇒ position setpoint = actual position value – The controller enable on the drive is reset – The following interface signals are reset: DB31, ... DBX61.5 = 0 (position controller active) DB31, ... DBX61.6 = 0 (speed controller active) DB31, ... DBX61.7 = 0 (current controller active) • Axis/spindle traverses – The axis is stopped with rapid stop. – Alarm 21612 "Controller enable VDI signal reset during motion". – The position control loop of the axis/spindle is opened. – Independent of the interface signal DB31, ... DBX1.4 (follow-up mode), the position setpoint is corrected at the end of the braking operation (position setpoint = actual position value) and the feedback signal DB31, ... DBX61.3 = 1 (follow-up mode) is set. – The following interface signals are reset: DB31, ... DBX61.5 (position controller active) DB31, ... DBX61.6 (speed controller active) DB31, ... DBX61.7 (current controller active)

DB31, ... DBX2.1	Controller enable
Application example(s)	Mechanical clamping of an axis If the axis is positioned at the clamping position, the clamping is closed. The controller enable is then reset. Otherwise, the position controller would constantly work against the clamping, if the axis was moved mechanically from its specified position during the clamping operation. When the clamping is removed, the controller enable is set first and then the mechanical clamping is released.
Special cases, errors, ...	Travel request for an axis/spindle without controller enable: • The axis/spindle is not traversed • The travel command is output to the interface • As long as the travel request is present, the axis/spindle is traversed immediately when the controller enable is set. A reset of the controller enable for a moving geometry axis always leads to a contour violation.
Corresponding to ...	DB31, ... DBX61.3 (follow-up active) DB31, ... DBX1.4 (follow-up mode) DB31, ... DBX61.5 (position controller active) DB31, ... DBX61.6 (speed controller active) DB31, ... DBX61.7 (current controller active) MD36620 $MA_SERVO_DISABLE_DELAY_TIME (OFF delay of the controller enable) MD36610 $MA_AX_EMERGENCY_STOP_TIME (braking ramp time when errors occur)

DB31, ... DBX2.2	Delete distance-to-go (axis-specific) / spindle reset
Edge evaluation: Yes	Signal(s) updated: Cyclically
Edge change 0 → 1	Delete distance-to-go (axis-specific) is requested. The behavior varies depending on the operating mode: • JOG mode: If the interface signal is set for one axis (edge change 0 → 1), this axis is stopped with ramp stop and its distance-to-go deleted (setpoint - actual value difference). Any remaining following error is still corrected. • Operating modes AUTOMATIC and MDI: The rising edge of the interface signals only influences the axes which are not in the geometry grouping. They are stopped with ramp stop and their distance-to-go deleted (setpoint - actual value difference). The next program block can then be started. IS "delete distance-to-go axial" is therefore ignored by geometry axes. Remark: "Delete distance-to-go" does not influence the running dwell time in a part program block with dwell time.
Edge change 1 → 0	No effect
Application example(s)	To terminate motion because of an external signal (e.g. measuring probe)

DB31, ... DBX2.2	**Delete distance-to-go (axis-specific) / spindle reset**
Special cases, errors, ...	"Delete distance-to-go (axis-specific)" After the axes have been stopped with "Delete distance-to-go" the next program block is prepared with the new positions. The axes thus follow a different contour to the one originally defined in the part program after a "Delete distance-to-go". If `G90` is programmed in the block after "Delete distance-to-go" it is at least possible to approach the programmed absolute position. On the other hand, with `G91`, the position originally defined in the part program is no longer reached in the following block.
Corresponding to ...	DB21, ... DBX6.2 (delete distance-to-go, channel-specific)
Additional references	Function Manual, Basic Functions; Spindles (S1)

DB31, ... DBX9.0 - DBX9.2	**Position controller parameter set selection A, B, C**			
Edge evaluation: No	Signal(s) updated: On request			
Meaning	Six different position controller parameter sets can be selected with the interface signals A, B, and C. The following assignment applies:			
	Parameter set	**C**	**B**	**A**
	1	0	0	0
	2	0	0	1
	3	0	1	0
	4	0	1	1
	5	1	0	0
	6	1	0	1
	6	1	1	0
	6	1	1	1
Signal irrelevant for ...	MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0			
Corresponding to ...	DB31, ...DBX69.0, .1, ..2 (feedback: Active position controller parameter set)			
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive, Section "Commissioning NCK" >			
Edge evaluation: No	Signal(s) updated: On request			

DB31, ... DBX9.3	**Parameter set switchover commands from NC disabled**
Edge evaluation: No	Signal(s) updated: On request
Signal state 1	The parameter set switchover is disabled.
Signal state 0	The parameter set switchover is enabled.
Corresponding to ...	DB31, ... DBX9.0, .1, .2

DB31, ... DBX20.1	**Ramp-function generator disable**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	A rapid stop with speed setpoint 0 is requested for the drive. The drive is stopped without a ramp function (regenerative braking). Feedback for the triggering of the rapid stop in the drive is via: DB31, ... DBX92.1 == 1 (ramp-function generator disable active)
Signal state 0	No rapid stop with speed setpoint 0 is requested for the drive.
Corresponding to ...	DB31, ... DBX92.1 (ramp-function generator disable active)
Remark	A PROFIDrive telegram compatible with the "SIMODRIVE 611 universal" interface mode must be set in the drive. See drive parameters: p0922 and p2038
Additional references	NC: Commissioning Manual IBN CNC: NCK, PLC, Drive Drive: SINAMICS S120/S150 List Manual

DB31, ... DBX21.0 - DBX21.2	**Drive parameter set selection A, B, C**
Edge evaluation: No	Signal(s) updated: Cyclically
Meaning	Eight different drive parameter sets can be selected with the interface signals A, B, and C. The following assignment applies:

Parameter set	C	B	A
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Application example(s)	Drive parameter switchover can be used for the following: • Changing the gear stage • Changing the measuring circuit
Special cases, errors, ...	In principle it is possible to switch over drive parameter sets at any time. However, as torque jumps can occur when switching over speed controller parameters and motor speed normalization, parameters should only be switched over in stationary states, especially axis standstill.
Corresponding to ...	DB31, ... DBX93.0, .1, .2 (feedback: Active drive parameter set)
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

<table>
<tr><td>DB31, ...
DBX21.3 - DBX21.4</td><td colspan="5">Motor selection A, B</td></tr>
<tr><td colspan="2">Edge evaluation: No</td><td colspan="4">Signal(s) updated: Cyclically</td></tr>
<tr><td>Meaning</td><td colspan="5">Switchover between four different motors or motor operating modes is possible with the interface signals A, B. The following assignment applies:
<table><tr><th>Motor selection</th><th>Application</th><th>B</th><th>A</th></tr><tr><td>Motor 1</td><td>Operating mode 1</td><td>0</td><td>0</td></tr><tr><td>Motor 2</td><td>Operating mode 2</td><td>0</td><td>1</td></tr><tr><td>Motor 3</td><td>Operating mode 3</td><td>1</td><td>0</td></tr><tr><td>Motor 4</td><td>Operating mode 4</td><td>1</td><td>1</td></tr></table>As soon as the request to switch over to a new motor or operating mode is detected in the drive, the pulse enable is removed.</td></tr>
<tr><td>Application example(s)</td><td colspan="5">For example, it is possible to switch between operating mode 1 (star operation) and operating mode 2 (delta operation) for a main spindle drive (MSD) via the motor selection.</td></tr>
<tr><td>Special cases, errors, ...</td><td colspan="5">Notice
Before a new selection, the following interface signal must be reset:
DB31, ... DBX21.5 = 0 (motor being selected)</td></tr>
<tr><td>Corresponding to ...</td><td colspan="5">DB31, ... DBX93.3 and .4 (feedback: Active motor)
DB31, ... DBX21.5 (motor being selected)</td></tr>
<tr><td>Additional references</td><td colspan="5">Commissioning Manual IBN CNC: NCK, PLC, Drive</td></tr>
</table>

DB31, ... DBX21.5	**Motor being selected**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The necessary electrical (e.g. contactor changeover for star-delta changeover) and/or mechanical changeovers are complete. The axis can be traversed again. The pulses are then enabled by the drive. **Notice** The interface signal must be reset before a new motor selection via DB31, ... DBX21.3 and .4.
Signal state 0	The necessary electrical (e.g. contactor changeover for star-delta changeover) and/or mechanical changeovers are **not** complete. The axis must not traverse. The pulses are not enabled by the drive.
Corresponding to ...	DB31, ... DBX93.3 and 4 (active motor) DB31, ... DBX21.3 and 4 (motor selection A, B)
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX21.6	**Integrator disable, speed controller**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The integrator (I component) of the speed controller is disabled or is to be disabled (P instead of PI behavior). **Note:** If the speed controller integrator disable is activated, compensation operations might take place in certain applications (e.g. if the integrator was already holding a load while stationary). The drive acknowledges the integrator disable to the PLC using the interface signal: DB31, ... DBX93.6 (speed controller integrator disabled).
Signal state 0	The integrator (I component) of the speed controller is enabled (PI behavior).
Corresponding to ...	DB31, ... DBX93.6 (feedback: The speed controller integrator is disabled)
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX21.7	**Pulse enable**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The pulses are enabled for the drive. The pulse enable is only performed in the drive when the drive signals readiness: DB31, ... DBX93.5 == 1 (feedback: Drive ready)
Signal state 0	The pulses are disabled for the drive.
Application example(s)	Signal relevant to safety
Special cases, errors, ...	If the pulse enable is removed during motion (e.g. emergency stop), the axis/spindle is no longer braked under control. The axis coasts to rest.
Corresponding to ...	DB31, ... DBX93.5 (feedback: Drive ready) DB31, ... DBX93.7 (feedback: Pulses are enabled)
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

18.1.9 Signals from axis/spindle (DB31, ...)

Note

SINAMICS S120: Message word (MELDW)

The message word (MELDW) is only contained in PROFIdrive telegrams compatible with SIMODRIVE 611U, for example, telegrams 102, 103, 105, 106, 110, 111, 116, 118, 125, 126, 136, 138, 139

References:
SINAMICS List Manual, Function Diagrams 2419 and 2420

SINAMICS S120: Status word 1/2 (ZSW1/2)

The status words ZSW1 or ZSW2 only refer to SIMODRIVE 611U compatible PROFIdrive telegrams in the following (SIMODRIVE 611U interface mode, p2038 = 1)

DB31, ... DBX61.0	**Drive test travel request**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The control signals that all of the traversing conditions for the drives are fulfilled. Requirements for this are: • The mechanical brake of the axis involved was previously released and all other axis traversing conditions are fulfilled. With: DB31, ... DBX61.0 (drive test, travel request) = 1 signal the appropriate axes can be moved. • The axis disable: DB31, ... DBX1.3 (axis/spindle disable) = 1 signal is not active.
Signal state 0	The control signals that the axes cannot be moved. Axes cannot be moved when: • DB31, ... DBX61.0 (drive test, travel request) = 0 signal • Faults are present in the control This means that the requirements specified above are not fulfilled.
Further references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX61.3	Follow-up active (feedback)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The control signals that the follow-up mode for the axis/spindle is not active. Requirements for this are: • The controller enable for the drive has been withdrawn (either by the PLC with "controller enable " = 0 signal or inside the control for faults; refer to the references) • Follow-up operation is selected (either by the PLC with IS "follow-up operation" = 1 signal or in the control, e.g. when withdrawing the controller enable from an axis that is moving) The position setpoint continually tracks the actual value while the follow-up mode is active. Zero speed and clamping monitoring are not active.
Signal state 0	The control signals that follow-up mode for the axis/spindle is not active. Zero speed and clamping monitoring are active. This means that the requirements specified above are not fulfilled. In the "hold" state, the interface signal: DB31, ... DBX61.3 (follow-up active) is 0.
Special cases, errors, ...	Notice: A delete distance-to-go is triggered internally in the control on transition from "Follow up" to "Hold" (IS "Follow-up mode" = 0) or in the closed-loop control mode (IS "Controller enable" = 1).
Corresponding to ...	DB31, ... DBX2.1 (controller enable) DB31, ... DBX1.4 (follow-up mode)
Additional references	Diagnostics Manual

DB31, ... DBX61.4	Axis/spindle stationary ($n < n_{min}$) (status)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The current speed of the axis or the actual number of rotations of the spindle lies below the limit given by the machine data: MD36060 $MA_STANDSTILL_VELO_TOL (maximum velocity/speed for signal "Axis/spindle stationary").
Signal state 0	The current velocity of the axis or the actual spindle speed is greater than the value specified in the MD (standstill range). If a travel command is present, e.g. for a spindle, then the signal is always = 0 - even if the actual speed lies below that specified in MD36060. If the interface signal: DB31, ... DBX61.4 (axis/spindle stationary) is signaled and there is no closed-loop position control active for the spindle, then at the MMC, an actual speed of zero is displayed and with the system variable $AA_S[n] zero is read.

DB31, ... DBX61.4	**Axis/spindle stationary (n < n_{min}) (status)**
Application example(s)	Enable signal for opening a protective device (e.g. open door). The workpiece chuck or the tool clamping device is only opened when the spindle is stationary. The oscillation mode can be switched-on during gear stage change after the spindle has been braked down to standstill. The tool clamping device must have been closed before the spindle can be accelerated.
Corresponding to ...	MD36060 $MA_STANDSTILL_VELO_TOL (maximum velocity/speed for signal "Axis/spindle stationary")

DB31, ... DBX61.5	**Position controller active (status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The control signals that the position controller for the axis or spindle is closed.
Signal state 0	The control signals that the position controller for the axis or spindle is open. If "controller enable" is canceled because of a fault or from the PLC user program the position controller is opened and therefore the interface signal DB31, ... DBX61.5 (position controller active) is set to 0. Spindle without position control: Signal "Position controller active" is always "0". See References for other effects.
Application example(s)	If the position control is active the axis/spindle is kept in position by the position controller. Any brakes or clamps can thus be opened. The interface signal: DB31, ... DBX61.5 (position controller active) can be used as feedback signal for the interface signal: DB31, ... DBX2.1 (controller enable). The holding brake of a vertical axis must be activated as soon as the position control is no longer active. If a spindle has been technically designed/dimensioned for the purpose, in the part program, it can be changed-over into the closed-loop position controlled mode as spindle or as axis (with `SPCON` or `M70`). In these cases, the interface signal "position controller active" is set.
Special cases, errors, ...	Special case for simulation axes (MD30350 $MA_SIMU_AX_VDI_OUTPUT = "1"): The IS "position controller active" is also set for simulation axes as soon as MD = "1".
Corresponding to ...	DB31, ... DBX2.1 (controller enable) DB31, ... DBX1.4 (follow-up mode) DB31, ... DBX1.5 and 1.6 (position measuring system 1 and 2)
Additional references	Diagnostics Manual

DB31, ... DBX61.6	**Speed controller active (status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The control signals that the speed controller is closed for the axis or spindle.
Signal state 0	The control signals that the speed controller is open for the axis or spindle. The speed controller output is cleared.
Application example(s)	If the spindle is not under position control, the interface signal can be used as a feedback for the interface signal DB31, ... DBX2.1 (controller enable).
Special cases, errors, ...	The interface signal is also set for simulation axes: Simulation axis: MD30350 $MA_SIMU_AX_VDI_OUTPUT == 1
Corresponding to ...	DB31, ... DBX61.5 (position controller active)

DB31, ... DBX61.7	**Current controller active (status)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The current controller for the axis/spindle is active.
Signal state 0	The current controller for the axis/spindle is not active. The current controller output, including the injection variables on the control voltage, is cleared.
Corresponding to ...	DB31, ... DBX61.5 (position controller active) DB31, ... DBX61.6 (speed controller active)

DB31, ... DBX69.0 - DBX69.2	**Active position controller parameter set A, B, C (feedback)**
Edge evaluation: No	Signal(s) updated: After switchover
Meaning	The active position controller parameter sets are displayed with the interface signals A, B and C. The following assignment applies:

Parameter set	C	B	A
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
6	1	1	0
6	1	1	1

Signal irrelevant for ...	MD35590 $MA_PARAMSET_CHANGE_ENABLE == 0 If the switchover of the position controller parameter set is switched off, the first parameter set is always active.
Corresponding to ...	DB31, ...DBX9.0 - DBX9.2 (position controller parameter set selection A, B, C)

DB31, ... DBX76.0	**Lubrication pulse**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Edge change 0 → 1 or 1 → 0	As soon as the axis/spindle has moved through the traversing distance set in machine data: MD33050 $MA_LUBRICATION_DIST (traversing distance for lubrication from the PLC), the interface signal is inverted. Note: The distance measurement is restarted each time that the control runs up.
Application example(s)	The lubrication pump for the axis/spindle can be activated with the "Lubrication pulse" interface signal. Machine bed lubrication therefore depends on the distance traveled.
Corresponding to ...	MD33050 $MA_LUBRICATION_DIST (lubrication pulse distance)

DB31, ... DBX92.1	**Ramp-function generator disable active (feedback)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Ramp function generator fast stop is active for the drive.
Signal state 0	Ramp function generator fast stop is not active for the drive.
Application example(s)	Position controller bypassing of the ramp function generator.
Corresponding to ...	DB31, ... DBX20.1 (ramp-function generator fast stop)
Remark	In the SINAMICS a protocol can be set that runs in what is known as the 611U compatibility mode.
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX92.4	**Drive-autonomous motion active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	A drive-autonomous motion is active. The axis traverses due to setpoints created by drive-internal functions. The drive still responds to control signals of the NC, e.g. controller enable. Setpoint specifications of the NC are ignored.
Signal state 0	A drive-autonomous motion is not active.
Application example(s)	Internal drive functions: • Rotor or pole position identification • Function generator
Remark	DBX92.4 = 1 **IF** MELDW.11 == 1 (controller enable) **AND** ZSW1.2 == 0 (operation enabled)
Additional references	SINAMICS S120 Function Manual; Pole position identification: Section "Servo control" > "Pole position identification" Function generator: Section "Servo control" > "Optimization of the current and speed controller"

DB31, ... DBX93.0 - DBX93.2	**Active drive parameter set A, B, C (feedback)**
Edge evaluation: No	Signal(s) updated: Cyclically
Meaning	The active drive parameter set is displayed with the interface signals A, B and C. The following assignment applies:

Parameter set	C	B	A
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Corresponding to ...	DB31, ... DBX21.0 - DBX21.2 (drive parameter set selection)
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX93.3 - DBX93.4	**Active motor A, B (feedback)**
Edge evaluation: No	Signal(s) updated: Cyclically
Meaning	Feedback signal from drive indicating which motor selection is active. The active motor or motor operating mode is displayed with the interface signals A, B. The following assignment applies:

Motor selection	Meaning	B	A
Motor 1	Star operation active	0	0
Motor 2	Delta operation active	0	1
Motor 3	Reserved	1	0
Motor 4	Reserved	1	1

Corresponding to ...	DB31, ... DBX21.3 and .4 (motor selection) DB31, ... DBX21.5 (motor being selected)
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX93.5	**Drive ready**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The drive is ready.
Signal state 0	The drive is not ready. If the signal is reset in operation, the drive is stopped (pulse inhibit or fast stop). When powering-up, the pulses are still inhibited (canceled) In addition, the following NC/PLC interface signals are reset: DB10, DBX108.6 = 0 (drive ready) DB31, ... DBX61.7 = 0 (current controller active) DB31, ... DBX61.6 = 0 (speed controller active)
Remark	DB31, ... DBX93.5 = MELDW.12
Corresponding to ...	**SINUMERIK** DB10, DBX108.6 (drive ready) DB31, ... DBX61.7 (current controller active) DB31, ... DBX61.6 (speed controller active)
Additional references	SINAMICS S120/S150 List Manual Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX93.6	**Speed controller integrator disabled (feedback)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The requested shutdown of the speed controller integrator is active in the drive. The speed controller has been switched over from the PI to P control response.
Signal state 0	The integrator of the speed controller is enabled. The speed controller functions as a PI controller.
Corresponding to ...	DB31, ... DBX21.6 (integrator disable, n-controller)
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX93.7	Pulses enabled (feedback)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The pulses of the assigned drive are enabled for the axis/spindle.
Signal state 0	The pulses of the assigned drive are not enabled. If the pulses on the assigned drive are deleted, the following interface signals are reset: • DB31, ... DBX61.7 (current controller active) • DB31, ... DBX61.6 (speed controller active) • DB31, ... DBX61.5 (position controller active)
Remark	DB31, ... DBX93.7 = MELDW.13
Corresponding to ...	DB31, ... DBX21.7 (pulse enable)
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX94.0	Motor temperature prewarning
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The motor temperature has exceeded the warning threshold (p0604) configured in the drive. Note If the motor temperature remains too high for longer than the parameterized time (p0606), a fault is output, the drive is stopped and the pulse enable removed. If the motor temperature falls below the warning threshold (p0604) again before the time has expired (p0606), the interface signal is reset.
Signal state 0	The motor temperature is below the warning threshold (p0604).
Remark	The current motor temperature is displayed on the user interface at: Operating area "Diagnostics" > "Service display: Axis/spindle"
Corresponding to ...	DB31, ... DBX94.1 (heat sink temperature prewarning)
Additional references	See DB31, ... DBX94.1 (heat sink temperature prewarning)

DB31, ... DBX94.1	**Heat sink temperature prewarning**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The heat sink temperature of the power semiconductors has exceeded the parameterized warning threshold (p0294). **Note** The parameterized reaction (p0290) is performed in the drive. If the temperature violation remains, a fault is output after approx. 20 s, the drive is stopped and the pulse enable removed.
Signal state 0	The drive module heat sink temperature monitoring has not responded, i.e. the temperature is below the warning threshold.
Remark	The interface signals DB31, ... DBX94.0 and .1 are derived from the following signals of the cyclic drive telegram: • Case 1: Temperature warning in the message word – DB31, ... DBX94.0 ≙ MELDW, bit 6 (no motor overtemperature warning) – DB31, ... DBX94.1 ≙ MELDW, bit 7 (no thermal overload in power unit warning) • Case 2: Warning of warning class B (interface mode "SIMODRIVE 611U", p2038 = 1) DB31, ... DBX94.0 == 1 and DBX94.1 == 1, if the following applies: Cyclic drive telegram, ZSW1: Bit 11 == 0 and 12 == 1 (warning class B) The interface signals are derived from the warning of warning class B if there is no specific information from the message word. An alarm is displayed. Alarm number = 200.000 + alarm value (r2124)
Additional references	• S120 Commissioning Manual, Section "Commissioning" > "Temperature sensors for SINAMICS components" • S120 Function Manual, Section "Monitoring and protective functions" • S120 List Manual – MELDW, bit 6 ≙ BO: r2135.14 → function diagram: 2548, 8016 – MELDW, bit 7 ≙ BO: r2135.15 → function diagram: 2548, 2452, 2456, 8016 • SINUMERIK Diagnostics Manual

DB31, ... DBX94.2	**Run-up completed**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The actual speed value has reached the speed tolerance band specified via p2164 after a new speed setpoint specification and has not left the band for the duration of p2166. Any subsequent speed fluctuations, also outside the tolerance band, e.g. due to load changes, will not affect the interface signal.
Signal state 0	The run-up procedure is still active after the speed setpoint has been changed.
Corresponding to ...	DB31, ... DBX94.6 ("n_{act} = n_{set}") DB31, ... DBX94.3 ("$\|M_D\|$ = M_{dx}")
Additional references	• SINUMERIK Commissioning Manual IBN CNC: NCK, PLC, Drive • SIMATIC S120 List Manual

DB31, ... DBX94.3	$\|M_d\| < M_{dx}$
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The current torque utilization is below the torque utilization threshold (torque threshold 2, p2194). The run-up procedure is completed, the drive is in the steady state and the torque setpoint $\|M_d\|$ drive does not exceed the threshold torque M_{dx}. The torque threshold characteristic is speed-dependent. During the run-up, DB31, ... DBX94.3 ($\|M_d\| < M_{dx}$) == 1. The interface signal is not updated until the run-up has been completed (DB31, ... DBX94.2 == 1) and the signal interlocking time for the threshold torque has expired.
Signal state 0	The torque setpoint $\|M_d\|$ is larger than the threshold torque M_{dx}. A motor overload can be determined via the interface signal. An appropriate response can then be initiated in the PLC user program.
Remark	DB31, ... DBX94.3 = MELDW.1
Additional references	• SINUMERIK Commissioning Manual IBN CNC: NCK, PLC, Drive • SIMATIC S120 List Manual

DB31, ... DBX94.4	$\|n_{act}\| < n_{min}$
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The actual speed value n_{act} is less than n_{min} (speed threshold value 3, p2161).
Signal state 0	The actual speed value is greater than the threshold minimum speed n_{min}.
Remark	DB31, ... DBX94.4 = MELDW.2
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX94.5	$\|n_{act}\| < n_x$
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The actual speed value n_{act} is less than n_x (speed threshold value 2, p2155).
Signal state 0	The actual speed value n_{act} is greater than the threshold speed n_x.
Remark	DB31, ... DBX94.5 = MELDW.3
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX94.6	$n_{act} = n_{set}$
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The actual speed value is at least the parameterized time (switch-on delay n_act = n_set, p2167) within the tolerance band around the speed setpoint (speed threshold value 4, p2163).
Signal state 0	The actual speed value is outside the tolerance band around the speed setpoint (speed threshold value 4, p2163).
Additional references	Commissioning Manual IBN CNC: NCK, PLC, Drive

DB31, ... DBX94.7	Variable signaling function
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The parameterized drive variable has exceeded the specified threshold value, including hysteresis.
Signal state 0	The parameterized drive variable has fallen below the specified threshold value, including hysteresis.
Remark	Using the "Variable signaling" function, BICO interconnections and parameters which have the attribute traceable can be monitored in the drive. DB31, ... DBX94.7 = MELDW.5
Additional references	SINAMICS S120 Function Manual, Section "Servo control" > "Variable signaling function"

DB31, ... DBX95.7	Warning of warning class C is pending
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The drive signals that a warning of warning class C is pending.
Signal state 0	The drive signals that no warning of warning class C is pending.
Remark	In the drive, a warning is the response to a detected potential or expected fault condition that does not cause the drive to switch off and does not have to be acknowledged.
Additional references	SINAMICS S120 List Manual, Section "Faults and alarms"

18.2 Axis monitoring, protection zones (A3)

18.2.1 Signals to channel (DB21, ...)

DB21, ... DBX1.1	Enable protection zones
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	When a positive edge of this signal appears, a protection zone is enabled and the active alarm cleared. Then, motion can start in the same protection zone. As a result of the start of motion, the protection zone is enabled, the IS "machine or channel-specific protection zone violated" is set, and the axis starts to move. The enabling signal is canceled if motion is started that does not lead into the enabled protection zone.
Signal state 0 or edge change 1 → 0	No effect
Application example(s)	This allows protection zones to be released: • If the current position is within a protection zone (alarm 2 present) • If motion is to be started towards the protection zone limit (alarm 1 or 2 present)

DB21, ... DBX8.0 - DBX9.1	Activate machine-specific protection zone 1 (...10)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The pre-activated, machine-related protection zone 1 (...10) is activated by the PLC user program. The protection zone is immediately activated. Only protection zones that have been pre-activated in the part program can be activated.
Signal state 0 or edge change 1 → 0	The pre-activated, machine-related protection zone 1 (...10) is de-activated by the PLC user program. The protection zone is immediately de-activated. Only protection zones that have been activated via the PLC and have been pre-activated in the NC part program can be de-activated.
Application example(s)	Before a sensor, for example, is moved into the working range, the relevant machine-related protection zone can be activated.

DB21, ... DBX10.0 - DBX11.1	Activate channel-specific protection zone 1 (...10)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The preactivated, channel-specific protection zone 1 (...10) is activated by the PLC user program. The protection zone is immediately activated. Only protection zones that have been pre-activated in the part program can be activated.
Signal state 0 or edge change 1 → 0	The pre-activated, channel-specific protection zone 1 (...10) is de-activated by the PLC user program. The protection zone is immediately de-activated. Only protection zones that have been activated via the PLC and have been pre-activated in the NC part program can be de-activated.
Application example(s)	Before a synchronous spindle, for example, is moved into the working range, the relevant machine-related protection zone can be activated.

18.2.2 Signals from channel (DB21, ...)

DB21, ... DBX272.0 – DBX273.1	Machine-related protection zone 1 (...10) pre-activated
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The machine-related protection zone 1 (...10) is preactivated in the current block. (Pre-activated in the part program). The protection zone can therefore be activated or de-activated in the PLC user program using the interface signal: DB21, ... DBX8.0 - DBX9.1 (machine-related protection zone 1 (...10))
Signal state 0 or edge change 1 → 0	The machine-related protection zone 1 (...10) is deactivated in the current block. (De-activated in the part program). The protection zone can therefore not be activated or de-activated in the PLC user program using the interface signal: DB21, ... DBX8.0 to DBX9.1 (activate machine-related protection zone 1 (...10))
Corresponding to	DB21, ... DBX8.0 - DBX9.1 (activate machine-related protection zone 1 (...10))

DB21, ... DBX274.0 – DBX275.1	**Channel-specific protection zone 1 (...10) pre-activated**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The channel-specific protection zone 1 (...10) is preactivated in the current block. (Pre-activated in the part program). The protection zone can therefore be activated or de-activated in the PLC user program using the interface signal: DB21, ... DBX10.0 - DBX11.1 channel-specific protection zone 1 (...10))
Signal state 0 or edge change 1 → 0	The channel-specific protection zone 1 (...10) is deactivated in the current block. (De-activated in the part program.) The protection zone can therefore not be activated or de-activated in the PLC user program using the interface signal: DB21, ... DBX10.0 - DBX11.1 (channel-specific protection zone 1 (...10))
Corresponding to	DB21, ... DBX10.0 - DBX11.1 (activate channel-specific protection zone 1 (...10))

DB21, ... DBX276.0 – DBX277.1	**Machine-related protection zone 1 (...10) violated**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The activated, machine-related protection zone 1 (...10) is violated in the current block or in the current JOG movement. The pre-activated, machine-related protection zone 1 (...10) would be violated in the current block if it would be activated by the PLC.
Signal state 0 or edge change 1 → 0	The activated, machine-related protection zone 1 (...10) is not violated in the current block. The pre-activated, machine-related protection zone 1 (...10) would not be violated in the current block if it would be activated by the PLC.
Application example(s)	Before parts are moved into the working zone - this IS can be used to check as to whether the tool or workpiece is located in the machine-related protection zone of the part to be moved in.

DB21, ... DBX278.0 - DBX279.1	**Channel-specific protection zone 1 (...10) violated**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The activated, channel-specific protection zone 1 (...10) is violated in the current block. The pre-activated, channel-specific protection zone 1 (...10) would be violated in the current block if it would be activated by the PLC.
Signal state 0 or edge change 1 → 0	The activated, channel-specific protection zone 1 (...10) is not violated in the current block. The pre-activated, channel-specific protection zone 1 (...10) would not be violated in the current block if it would be activated by the PLC.
Application example(s)	Before parts are moved into the working zone - this IS can be used to check whether the tool or workpiece is located in the channel-specific protection zone of the part to be moved in.

18.2.3 Signals to axis/spindle (DB31, ...)

DB31, ... DBX2.3	**Clamping in progress**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Clamping in progress. The clamping monitoring function is activated.
Signal state 0 or edge change 1 → 0	Clamping completed. The clamping monitoring function is replaced by the standstill (zero speed) monitoring.
Corresponding to	MD36050 $MA_CLAMP_POS_TOL (Clamping tolerance)

DB31, ... DBX3.6	**Velocity/spindle speed limitation**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The NCK limits the velocity/spindle speed to the limit value set in the machine data: MD35160 $MA_SPIND_EXTERN_VELO_LIMIT
Signal state 0 or edge change 1 → 0	No limitation active.
Corresponding to	MD35100 $MA_SPIND_VELO_LIMIT (max. spindle speed) SD43220 $SA_SPIND_MAX_VELO_G26 (prog. spindle speed limiting G26) MD43230 $SA_SPIND_MAX_VELO_LIMIT (prog. spindle speed limiting G96/G961)

DB31, ... DBX12.0 - DBX12.1	**Hardware limit switches plus and minus**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	A switch can be mounted at each end of the travel range of a machine axis which will cause a signal "hardware limit switch plus or minus" to be signaled to the NC via the PLC if it is actuated. If the signal is recognized as set, alarm 021614 "hardware limit switch + or -" is output and the axis is decelerated immediately. The braking/deceleration type is defined using the machine data: MD36600 $MA_BRAKE_MODE_CHOICE (braking behavior at the hardware limit switch) If the controller enable is withdrawn at the same time as the "hardware limit switch" signal, then the axis responds as described in Chapter A2 ("various interface signals").
Signal state 0 or edge change 1 → 0	Normal condition - a hardware limit switch has not been actuated.
Corresponding to	MD36600 $MA_BRAKE_MODE_CHOICE (deceleration behavior when the hardware limit switch responds)

DB31, ... DBX12.2 - DBX12.3	**2nd software limit switch plus or minus**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	2nd software limit switch for the plus and minus directions is effective. 1st software limit switch for the plus and minus directions is not effective. In addition to the 1st software limit switches (plus and minus), the 2nd software limit switches (plus and minus) can be activated using these interface signals. The position is defined using machine data: MD36130 $MA_POS_LIMIT_PLUS2 (2nd software limit switch plus) and MD36120 $MA_POS_LIMIT_MINUS2 (2nd software limit switch minus) .
Signal state 0 or edge change 1 → 0	1st software limit switch for the plus and minus directions is effective. 2nd software limit switch for the plus and minus directions is not effective.
Corresponding to	MD36110 $MA_POS_LIMIT_PLUS (1st software limit switch plus) MD36130 $MA_POS_LIMIT_PLUS2 (2nd software limit switch plus) MD36100 $MA_POS_LIMIT_MINUS (1st software limit switch minus) MD36120 $MA_POS_LIMIT_MINUS2 (2nd software limit switch minus)

18.2.4 Signals from axis/spindle (DB31, ...)

DB31, ... DBX60.2 - DBX60.3	**Encoder limit frequency exceeded 1 Encoder limit frequency exceeded 2**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The limit frequency set in the machine data: MD36300 $MA_ENC_FREQ_LIMIT (encoder limit frequency) has been exceeded. The reference point for the position measuring system involved has been lost (IS: Referenced/synchronized has a signal state 0). Closed-loop position control is no longer possible. Spindles continue to run with closed-loop speed control. Axes are stopped with a fast stop (with open-circuit position control loop) along a speed setpoint ramp.
Signal state 0 or edge change 1 → 0	The limit frequency set in machine data: MD36300 $MA_ENC_FREQ_LIMIT is no longer exceeded. For the edge change 1 → 0, the encoder frequency must have fallen below the value of machine data: MD36302 $MA_ENC_FREQ_LIMIT_LOW (encoder limit frequency for encoder resynchronization)

18.3 Continuous-path mode, exact stop and LookAhead (B1)

18.3.1 Signals from channel (DB21, ...)

DB21, ... DBX36.3	All axes stationary
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	All axes assigned to the channel are stationary with interpolator end. No other traversing movements are active.

18.3.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX60.6	Position reached with exact stop coarse
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The axis is in the appropriate exact stop and no interpolator is active for the axis and: • The control is in the reset state (reset key or end of program). • The axis was last programmed as a positioning axis or positioning spindle (initial setting of supplementary axis: Positioning axis). • The path motion was terminated with NC stop. • The spindle is in the closed-loop positioncontrolled mode (SPCON/SPOS instruction) and is stationary. • The axis is switched from closed-loop speedcontrolled to closed-loop positioncontrolled mode with IS "position measuring system".
Signal state 0 or edge change 1 → 0	The axis is not in the appropriate exact stop or the interpolator is active for the axis or: • The path motion was terminated with NC stop. • The spindle is in the closed-loop speedcontrolled mode (SPCOF/SPOSA instruction). • The "followup" mode is active for the axis. • The "parking" mode is active for the axis. • The axis is switched from closed-loop positioncontrolled to closed-loop speedcontrolled mode with IS "position measuring system".
Signal irrelevant for ...	Rotary axes that are programmed as rounding axes.
Corresponding to ...	MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)

DB31, ... DBX60.7	**Position reached with exact stop fine**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Refer to DB31, ... DBX60.6 (position reached with exact stop coarse).
Signal state 0 or edge change 1 → 0	Refer to DB31, ... DBX60.6 (position reached with exact stop coarse).
Signal irrelevant for ...	Rotary axes that are programmed as rounding axes.
Corresponding to ...	MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)

18.4 Travel to fixed stop (F1)

18.4.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX1.1	**Acknowledge fixed stop reached**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Significance after the fixed stop has been reached: DB31, ... DBX62.5 (fixed stop reached) = 1 → The axis presses against the fixed stop with the clamping torque: → The fixed stop monitoring window is activated. → A block change is executed.
Signal state 0 Edge change 1 → 0	Significance after the fixed stop has been reached: DB31, ... DBX62.5 (fixed stop reached) = 1 → The axis presses against the fixed stop with the clamping torque. → The fixed stop monitoring window is activated. → A block change is not executed and the channel message "Wait: Auxiliary function acknowledgment missing" is displayed. Meaning after the fixed stop has been reached: IS "Fixed stop reached" DB31, ... DBX62.5 = 1 → The function is aborted, the alarm "20094 axis %1 Function aborted" is output. Significance when de-selecting the function `FXS=0` via the part program: → The torque limiting and the monitoring of the fixed stop monitoring window are canceled.
IS irrelevant for ...	MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgments for travel to fixed stop) = 0 or 1
Corresponding to	MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgments for travel to fixed stop) DB31, ... DBX62.5 (fixed stop reached)

DB31, ... DBX1.2	**Sensor for fixed stop**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Fixed stop is reached.
Signal state 0 or edge change 1 → 0	Fixed stop is not reached.
Corresponding to	The signal is only active if: MD37040 $MA_FIXED_STOP_BY_SENSOR = 1

DB31, ... DBX3.1	**Enable travel to fixed stop**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	Meaning when FXS function is selected using the part program (IS "Activate travel to fixed stop" = 1): → Travel to fixed stop is enabled and the axis traverses from the start position at the programmed velocity to the programmed target position.
Signal state 0	Meaning when FXS function is selected using the part program (IS "Activate travel to fixed stop" = 1): → Travel to fixed stop is inhibited. → The axis is stationary at the start position with reduced torque. → The channel message "wait": Auxiliary function acknowledgment missing" is displayed.
Edge change 1 → 0	Meaning before the fixed stop has been reached (IS "fixed stop reached" = 0): → Travel to fixed stop is aborted. → The alarm "20094: Axis%1 Function aborted" is displayed. Meaning after the fixed stop has been reached IS "fixed stop reached" = 1): → The torque limiting and monitoring of the fixed stop monitoring window are canceled. Deselection: DB31, ...DBX1.1 (acknowledge fixed stop reached)
IS irrelevant for ...	MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgments for travel to fixed stop) = 0 or 2
Corresponding to	MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgments for travel to fixed stop) DB31, ... DBX62.4 (activate travel to fixed stop)

18.4.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX62.4	Activate travel to fixed stop
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The "Travel to fixed stop" function is active. This signal is used for analog drives in order, for example, to activate the current or torque limitation parameterized in the actuator.
Signal state 0 or edge change 1 → 0	The "Travel to fixed stop function" is not active.

DB31, ... DBX62.5	Fixed stop reached
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The fixed stop was reached after selecting the FXS function. This signal is used by analog drives, e.g. to switch the actuator from speedcontrolled to current or torquecontrolled mode so that a programmable clamping torque can be set.
Signal state 0 or edge change 1 → 0	The fixed stop has still not been reached after selecting the FXS function.

18.5 Help function output to PLC (H2)

18.5.1 Signals to channel (DB21, ...)

DB21, ... DBX30.5	**Activate associated M01**
Edge evaluation: no	Signal(s) updated:
Signal state 1 or edge change 0 → 1	PLC signals the NCK that the associated `M01` (help function) should be activated.
Signal state 0 or edge change 1 → 0	De-activate the associated `M01` (help function).
Corresponding to	DB21, ... DBX 318.5 (associated `M01` active)

18.5.2 Signals from channel (DB21, ...)

DB21, ... DBB58, DBB60 - DBB65	**M, S, T, D, H, F functions Modification**
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	One M, S, T, An D, H, or F function has been output to the interface with a new value together with the associated change signal at the beginning of an OB1 cycle. In this case, the change signal indicates that the appropriate value is valid.
Signal state 0 or edge change 1 → 0	The change signals are reset by the PLC basic program at the start of the next OB1 cycle. The value of the data involved is not valid.

DB21, ... DBX59.0 - DBX59.4	**M fct. 1-5 not decoded**
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	M function is greater than 99 (for extended address = 0) or for extended address > 0, not included in the decoding list. This signal is available - together with the associated M change signal - for one OB1 cycle. Cause: • Incorrect M function programmed • M function not configured in the decoding list of the PLC Remedy, e.g.: • PLC sets read-in disable • Output of a PLC alarm
Signal state 0 or edge change 1 → 0	M function less than 99 (for extended address = 0) or for extended address > 0 included in the decoding list.

DB21, ... DBB60 - DBB64, DBB66 - DBB67	**M, S, T, D, H, F functions Additional info "Quick" (fast acknowledgment)**
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	One M, S, T, An D, H, or F function has been output to the interface with a new value together with the associated change signal at the beginning of an OB1 cycle. In this case, the additional info "Quick" indicates the quick help function.
Signal state 0 or edge change 1 → 0	The change signals are reset by the PLC basic program at the start of the next OB1 cycle. The value of the data involved is not valid.

DB21, ... DBB68 - DBB97	**M functions 1 to 5 Extended address M functions 1 to 5**
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	Up to 5 M functions programmed in an NC block are simultaneously made available here as soon as the M change signals are available. Value range of M functions: 0 to 9999 9999; integer number Value range of the extended address: 0 to 99; integer number The M functions remain valid until they are overwritten by new M functions.
Signal state 0 or edge change 1 → 0	• After the PLC has ramped-up. • All help functions are deleted before a new function is entered.
Application example(s)	Decoding and evaluation of M functions that are not decoded as standard or via a list. Using the extended address, the M function can be assigned to another channel that does not correspond to that channel in which the program is running.
Special cases, errors, ...	For M00 to M99 the extended address = 0.

DB21, ... DBB98 - DBB115	**S functions 1 to 3 Extended address S functions 1 to 3**
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	Up to 3 S functions programmed in an NC block are simultaneously made available here as soon as the S change signals are available. Value range of the spindle speed: 0 to 999 999; integer number Value range of the extended address: 0 to 6; integer number The S functions remain valid until they are overwritten by new S functions.
Signal state 0 or edge change 1 → 0	• After the PLC has ramped-up. • All help functions are deleted before a new function is entered.
Application example(s)	Spindle speed control by the PLC. The extended address is used to program for which spindle the S word is valid. E.g.: S2=500

DB21, ... DBB118	**T function 1**
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	The T function programmed in an NC block is made available here as soon as the T change signal is available. Value range of T functions: 0 to 99 999 999; integer number The T function remains valid until it is overwritten by a new T function.
Signal state 0 or edge change 1 → 0	• After the PLC has ramped-up. • All help functions are deleted before a new function is entered.
Application example(s)	Control of automatic tool selection.
Special cases, errors, ...	With `T0`, the current tool is removed from the tool holder but not replaced by a new tool (default configuration of the machine manufacturer).
Remark	8 decade T numbers are only available as T function 1.

DB21, ... DBB129	**D function 1**
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	The D function programmed in an NC block is made available here as soon as the D change signal is available. Value range of D functions: 0 to 999; integer number The D function remains valid until it is overwritten by a new D function.
Signal state 0 or edge change 1 → 0	• After the PLC has ramped-up. • All help functions are deleted before a new function is entered.
Application example(s)	Implementation of protective functions.
Special cases, errors, ...	`D0` is reserved for deselecting the current tool offset.

DB21, ... DBB140 - DBB157	H functions 1 to 3 Extended address H functions 1 to 3
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	Up to 3 H functions programmed in an NC block are made available here simultaneously as soon as the H change signals are available. Value range of the H function: Floating point (corresponding to the MC5+format) Value range of the extended address: 0 to 99; integer number The H functions remain valid until they are overwritten by new H functions.
Signal state 0 or edge change 1 → 0	• After the PLC has ramped-up. • All help functions are deleted before a new function is entered.
Application example(s)	Switching functions on the machine.

DB21, ... DBB158 - DBB193	F functions 1 to 6 Extended address F functions 1 to 6
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	Up to 6 F functions (one path feed and up to 5 axis-specific feeds for positioning axes) are made available here simultaneously as soon as the F change signals are available. Value range of F function: Floating point (corresponding to the MC5+format) Value range of the extended address: 0 to 18; integer number The extended address of the F function is generated from the feed type (path feed or axis-specific feed) and the axis names. It is coded as follows: 0: Path feed; e.g.: F=1000 — e.g.: F=1000 1 to 18: Machine axis number of the positioning axis for an axis-specific feed — e.g.: FA[X1]=500 The F functions remain until they are overwritten by new F functions.
Signal state 0 or edge change 1 → 0	• After the PLC has ramped-up. • All help functions are deleted before a new function is entered.
Application example(s)	Control of programmed F word by the PLC, e.g. through overwriting of the set feed rate override.
Corresponding to ...	MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

DB21, ... DBB194 - DBB206	**Dynamic M functions: M0 - M99**
Edge evaluation: No	Signal(s) updated: job-controlled by NCK
Signal state 1 or edge change 0 → 1	The dynamic M signal bits are set by decoded M functions.
Signal state 0 or edge change 1 → 0	For a general help function output, the dynamic M signal bits are acknowledged by the PLC basic program after the OB1 has been completely run-through (executed once). For a fast help function output, after the PLC identifies the help functions, they are acknowledged in the same OB40 cycle.
Application example(s)	Spindle clockwise/counterclockwise rotation, switch coolant ON/OFF

DB21, ... DBX318.5	**Associated M01/M00 active**
Edge evaluation: No	Signal(s) updated:
Signal state 1 or edge change 0 → 1	This bit indicates that an `M00` or `M01` help function is active if the appropriate associated `M00` and `M01` (help functions) were enabled/activated beforehand.
Signal state 0 or edge change 1 → 0	No associated `M00`/`M01` help functions active.
Corresponding to ...	DB21, ... DBX30.5 (activate associated `M01`)

18.5.3 Signals from axis/spindle (DB31, ...)

DB31, ... DBD78	F auxiliary function for positioning axis
Edge evaluation: no	Signal(s) updated: Jobcontrolled
	The values of the F help functions for positioning axes are stored here. The axis to which each value applies is determined by the extended address.

DB31, ... DBD86	M auxiliary function for spindle
Edge evaluation: no	Signal(s) updated: Jobcontrolled
	The values for the M3, M4, M5 help functions are sent to the associated interface for the addressed spindle.

DB31, ... DBD88	S auxiliary function for spindle
Edge evaluation: no	Signal(s) updated: Jobcontrolled
	The values for the S help functions are sent to the associated interface for the addressed spindle.

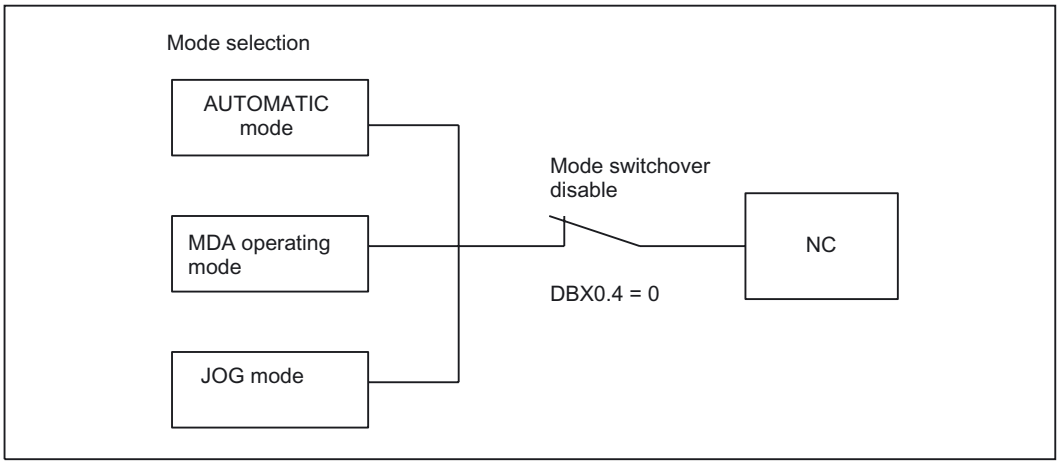
18.6 Mode group, channel, program operation, reset response (K1)

18.6.1 Signals to mode group (DB11)

DB11 DBX0.0	AUTOMATIC mode
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	AUTOMATIC mode is selected by the PLC program.
Signal state 0 or edge change 1 → 0	AUTOMATIC mode is not selected by the PLC program.
Signal irrelevant for ...	DB11 DBX0.4 (operating mode, changeover inhibit) = 1
Corresponding to ...	DB11 DBX6.0 (active AUTOMATIC mode)

DB11 DBX0.1	MDA mode
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	MDA mode is selected by the PLC program.
Signal state 0 or edge change 1 → 0	MDA mode is not selected by the PLC program.
Signal irrelevant for ...	DB11 DBX0.4 (operating mode, changeover inhibit) = 1
Corresponding to ...	DB11 DBX6.1 (active MDA mode)

DB11 DBX0.2	JOG mode
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	JOG mode is selected by the PLC program.
Signal state 0 or edge change 1 → 0	JOG mode is not selected by the PLC program.
Signal irrelevant for ...	DB11 DBX0.4 (operating mode, changeover inhibit) = 1
Corresponding to ...	DB11 DBX6.2 (active JOG mode)

DB11 DBX0.4	Mode change disable
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The currently active mode (JOG, MDA or AUTOMATIC) of the mode group cannot be changed. The machine functions that can be selected within a mode group can be changed.
Signal state 0 or edge change 1 → 0	The mode of the mode group can be changed.
	

DB11 DBX0.5	**Mode group stop**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	An NC stop is activated for all the channels of the mode group. The channel status of all the active channels changes to the channel status "interrupted". All of the channels in channel status "reset" remain in the channel status "reset". Programs that are running at this point are immediately interrupted (at the earliest possible point, even within a block) and the program status changes to "stopped". All the moving axes of the mode group are decelerated according to their acceleration characteristics without contour violation. The program can be restarted with NC start. None of the spindles of that mode group are affected.
Signal state 0 or edge change 1 → 0	Channel status and program execution are not influenced.
Special cases, errors, ...	All the axes of a mode group that are not triggered by a program or a program block (e.g. axes traverse because traverse keys are being pressed on the machine control panel) decelerate to rest with mode group stop.

DB11 DBX0.6	**Mode group stop axes plus spindles**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	An NC stop is activated for all the channels of the mode group. The channel status of all of the active channels changes to the channel status "interrupted". All of the channels in channel status "reset" remain in the channel status "reset". Programs that are running at this point are immediately interrupted (at the earliest possible point, even within a block) and the program status changes to "stopped". All the moving axes and spindles of the mode group are decelerated according to their acceleration characteristics without contour violation. The program can be restarted with NC start.
Signal state 0 or edge change 1 → 0	Channel status and program execution are not influenced.
Special cases, errors, ...	All the axes and spindles of a mode group that are not triggered by a program or a program block (e.g. axes traverse because traverse keys are pressed on the machine control panel, spindles are controlled by the PLC) decelerate to rest with "mode group stop plus spindles".

DB11 DBX0.7	**Mode group reset**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	A reset is activated for all the channels of the mode group. All of the channels are then in the channel status "reset". All of the current programs are then in the program status "aborted". All moving axes and spindles are decelerated to zero speed according to their acceleration ramp without contour violation. The initial settings are set (e.g. for G functions). The alarms for the mode group are cleared if they are not POWER ON alarms.
Signal state 0 or edge change 1 → 0	Channel status and program execution are not influenced by this signal.
Corresponding to ...	DB21, ... DBX7.7 (channel reset) DB11 DBX6.7 (all channels in the reset state)
Special cases, errors, ...	An alarm that withdraws the interface signal DB11 DBX6.3 (mode group ready) ensures that all channels of the mode group are no longer in the reset state. In order to switch to another operating mode, a mode group reset (DB11 DBX0.7) must then be initiated.

DB11 DBX1.0	**Machine function TEACH IN**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Machine function TEACH IN is activated in JOG mode for the mode group.
Signal state 0 or edge change 1 → 0	Machine function TEACH IN is not activated.
Signal irrelevant for ...	If JOG mode is not active.
Additional references	Operating Manual HMI (corresponding to the used software)

DB11 DBX1.1	Machine function REPOS
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Machine function REPOS is activated in JOG mode for the mode group.
Signal state 0 or edge change 1 → 0	Machine function REPOS is not activated.
Signal irrelevant for ...	JOG mode is not active.
Application example(s)	When a fault occurs when executing a part program (e.g. tool breakage), the axis is manually moved away from the fault location in the JOG mode in order to be able to replace the tool. The axis can then be manually returned to the exact previous position using the REPOS machine function so that the program can be continued in the automatic mode.
Additional references	Operating Manual HMI (corresponding to the used software)

DB11 DBX1.2	Machine function REF
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Machine function REF is activated in the JOG mode for the mode group.
Signal state 0 or edge change 1 → 0	Machine function REF is not activated.
Signal irrelevant for ...	If JOG mode is not active.
Additional references	See Section "R1: Referencing (Page 1319)"

DB11 DBX1.6	**Single block type B**
Edge evaluation: No	Signal(s) updated:
Signal state 1 or edge change 0 → 1	Bit set and DB11 DBX1.7 not set: Response across modes • All channels are stopped. • All channels receive a start command. • Channel KS stops at the end of the block. • The channels KA receive a STOPATEND. (comparable with DB21, ... DBX7.2 (NC stop at the block limit).) • All channels are stopped at a block limit (at some point in time). (If DB11 DBX1.6 and DB11 DBX1.7 are set simultaneously, it is impossible to determine which single block type is required. The control then assumes: No single block across mode groups).
Signal state 0 or edge change 1 → 0	If Bit DB11 DBX1.6 is not set and Bit DB11 DBX1.7 is set, then it is single block type A. (If DB11 DBX1.6 and DB11 DBX1.7 are not set, it is impossible to determine which single block type is required. The control then assumes: No single block across mode groups).
Corresponding to ...	Single block type A

DB11 DBX1.7	**Single block type A**
Edge evaluation: No	Signal(s) updated:
Signal state 1 or edge change 0 → 1	DB11 DBX1.7 set and DB11 DBX1.6 not set: Response across modes • All channels are stopped. • All channels receive a start (start key). • Channel KS stops at the end of the block (due to single-block) • Channels KA receive a STOP command. (analogous to STOP key) • All channels are stopped. (deceleration phase of all KAs) (If DB11 DBX1.6 and DB11 DBX1.7 are set simultaneously, it is impossible to determine which single block type is required. The control then assumes: No single block across mode groups).
Signal state 0 or edge change 1 → 0	If DB11 DBX1.7 is not set and DB11 DBX1.6 is set, then it is single block type B. (If DB11 DBX1.6 and DB11 DBX1.7 are not set, it is impossible to determine which single block type is required. The control then assumes: No single block across mode groups).
Corresponding to ...	Single block type B

18.6.2 Signals from the mode group (DB11)

DB11 DBX4.0	Selected mode AUTOMATIC
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	AUTOMATIC mode is selected by HMI.
Signal state 0 or edge change 1 → 0	AUTOMATIC mode is not selected by HMI.

DB11 DBX4.1	Selected mode MDA
Edge evaluation:	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	MDA mode is selected by HMI.
Signal state 0 or edge change 1 → 0	MDA mode is not selected by HMI.

DB11 DBX4.2 Data block	Selected JOG mode Signal(s) from BAG (HMI → PLC)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	JOG mode is selected by HMI.
Signal state 0 or edge change 1 → 0	JOG mode is not selected by HMI.

DB11 DBX5.0	**Selected machine function TEACH IN**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The machine function TEACH IN is selected by HMI within BAG.
Signal state 0 or edge change 1 → 0	The machine function TEACH IN is not selected by HMI.
Additional references	Operating Manual HMI (corresponding to the used software)

DB11 DBX5.1	**Selected REPOS machine function**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The machine function REPOS is selected by HMI within BAG.
Signal state 0 or edge change 1 → 0	The machine function REPOS is not selected by HMI.
Application example(s)	When a fault occurs when executing a part program (e.g. tool breakage), the axis is manually moved away from the fault location in the JOG mode in order to be able to replace the tool. The axis can then be manually returned to the exact previous position using the REPOS machine function so that the program can be continued in the automatic mode.
Additional references	Operating Manual HMI (corresponding to the used software)

DB11 DBX5.2	**Selected machine function REF**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The machine function REF is selected by HMI within BAG.
Signal state 0 or edge change 1 → 0	The machine function REF is not selected by HMI.
Additional references	See Section "R1: Referencing (Page 1319)"

DB11 DBX6.0 Data block	**Active mode AUTOMATIC Signal(s) from the mode group (NCK → PLC)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	AUTOMATIC mode is active.
Signal state 0 or edge change 1 → 0	AUTOMATIC mode is not active.

DB11 DBX6.1	**Active mode MDA**
Edge evaluation:	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	MDA mode is active.
Signal state 0 or edge change 1 → 0	MDA mode is not active.

DB11 DBX6.2	**Active JOG mode**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	JOG mode is active.
Signal state 0 or edge change 1 → 0	JOG mode is not active.

DB11 DBX6.3	**Mode group ready**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	This signal is set after power on and all of the voltage have been established. The mode group is now ready and part programs can be executed and axes traversed in the individual channels.
Signal state 0 or edge change 1 → 0	The mode group is not ready. Possible causes for this are: • A critical axis or spindle alarm is present • Hardware faults • The mode group has been incorrectly configured (machine data) If the mode group ready changes to signal state "0", then: • The axis and spindle drives are braked down to standstill with the max. braking current. • The signals from the PLC to the NCk are brought into an inactive state (cleared state).
Special cases, errors, ...	An alarm that withdraws the interface signal DB11 DBX6.3 (mode group ready) ensures that all channels of the mode group are no longer in the reset state. In order to switch to another operating mode, a mode group reset (DB11 DBX0.7) must be made.

DB11 DBX6.7	**All channels in the reset state**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	All the channels that belong to this mode group are in the "channel status reset" state (DB21, ... DBX7.7).
Signal state 0 or edge change 1 → 0	At least one of the channels in the mode group is not in "channel status reset" (DB21, ... DBX7.7).
Corresponding to ...	DB21, ... DBX7.7 (channel state, reset)

DB11 DBX7.0	**Active machine function TEACH IN**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Machine function TEACH IN is active in the mode group.
Signal state 0 or edge change 1 → 0	Machine function TEACH IN is not active.
Additional references	Operating Manual HMI (corresponding to the used software)

DB11 DBX7.1	Active REPOS machine function
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Machine function REPOS is active in the mode group.
Signal state 0 or edge change 1 → 0	Machine function REPOS is not active.
Application example(s)	When a fault occurs when executing a part program (e.g. tool breakage), the axis is manually moved away from the fault location in the JOG mode in order to be able to replace the tool. The axis can then be manually returned to the exact previous position using the REPOS machine function so that the program can be continued in the automatic mode.
Additional references	Operating Manual HMI (corresponding to the used software)

DB11 DBX7.2	Active machine function REF
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Machine function REF is active in the mode group.
Signal state 0 or edge change 1 → 0	Machine function REF is not active.
Additional references	See Section "R1: Referencing (Page 1319)"

18.6.3 Signals to channel (DB21, ...)

DB21, ... DBX0.4	**Activate single block**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	In AUTOMATIC and MDI modes, the operator must enable processing of each individual part program block of the part program selected in the channel by reactivating NC start.
Signal state 0	No effect.
Special cases, errors, ...	• In the case of active tool offset, intermediate blocks are inserted, when necessary. These blocks must also be enabled using NC start. • In a series of G33 blocks, a single block is only operative if "dry run feed" is selected. • In the case of a decoding single block, calculation blocks are not processed in the single step.
Corresponding to ...	DB21, ... DBX35.3 (program status interrupted)

DB21, ... DBX0.5	**Activate M01**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Activation of program control "Conditional stop"M01 is requested.
Signal state 0	Activation of program control "Conditional stop" M01 is not requested.
Corresponding to ...	DB21, ... DBX24.5 (M01 selected) DB21, ... DBX32.5 (M0/M01 active)

DB21, ... DBX1.6	**PLC action completed**
Edge evaluation: No	Signal(s) updated: Cyclically
	At the end of the block search, concluding action blocks are executed: DB21, ... DBX32.3 (action block active) == 1 AND DB21, ... DBX32.6 (last action block active) == 1 Alarm "10208 Channel <*Channel Number*> Issue NC start to continue program" notifies the user that he must reactivate NC start to resume the part program starting from the target block. If other actions are to be executed by the PLC user program prior to the NC start (e.g. tool change), Search mode can be parameterized as follows: MD11450 $MN_SEARCH_RUN_MODE = 1 Output of alarm delayed until the existing signal is reset.
Signal state 1	PLC action is completed.
Signal state 0	PLC action is not yet completed.
Corresponding to ...	DB21, ... DBX32.3 (action block active) DB21, ... DBX32.6 (last action block active) DB21, ... DBX33.4 (block search active)

DB21, ... DBX1.7	Activate program test
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Activation of the program test is requested. During the program test, all motion commands of axes (not spindles) take place under "Axis disable." **Notice!** Due to the axis disable, the assignment of a tool magazine is not changed for the program test. The user/machine manufacturer must utilize a suitable PLC user program to ensure that the NCK-internal tool management and the actual assignment of the tool magazine remain consistent. Refer to the program example included in the PLC Toolbox.
Signal state 0	Activation of the program test is not requested.
Corresponding to ...	DB21, ... DBX25.7 (program test selected) DB21, ...DBX33.7 (program test active)

DB21, ... DBB2	Activate skip block
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Skip blocks marked in the part program with an slash (/) are not processed. If there is a series of skip blocks, the signal is only active if it is present before the first skip block of the series is decoded. Note The signal should be available prior to the start of the part program.
Signal state 0	Skip blocks marked in the part program with an slash (/) are processed.
Corresponding to ...	DB21, ... DBX26.0 (skip block selected) DB21, ... DBX35.2 (program status stopped)

DB21, ... DBX6.1	Read-in disable
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The main run reads in no more preprocessed part program blocks. Note The signal is only active in AUTOMATIC and MDI modes.
Signal state 0	The main run reads in preprocessed part program blocks.
Corresponding to ...	DB21, ... DBX35.0 (program status running)

DB21, ... DBX6.4	**Program level abort**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Edge change 0 → 1	At each edge change 0 → 1 the current program level being processed (subprogram level, ASUB level, save routine) is immediately aborted. Processing of the part program continues at the next higher program level from the exit point.
Edge change 1 → 0	No effect.
Special cases, errors, ...	The main program level cannot be aborted with this IS, only with IS "Reset".

DB21, ... DBX7.0	**NC start disable**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The NC start disable prevents a part program from being started with NC START signal DB21, ... DBX7.1 (NC start) == 1.
Signal state 0	NC start disable is not active.
Special cases, errors, ...	The start of a part program selected in the channel by part program command `START` in another channel (program coordination) is not prevented by the interface signal: DB21, ... DBX7.0 (NC start disable) == 1.
Corresponding to ...	DB21, ... DBX7.1 (NC start)

DB21, ... DBX7.1	**NC start**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Edge change 0 → 1	AUTOMATIC mode: The selected NC program is started or continued, or the auxiliary functions that were saved during the program interruption are output. If data are transferred from the PLC to the NC during program status "Program interrupted," then these data are immediately cleared at NC start. MDI:mode The entered block information or part program blocks are released for execution.
Edge change 1 → 0	No effect.
Corresponding to ...	DB21, ... DBX7.0 (NC start disable)

DB21, ... DBX7.2	**NC stop at block limit**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The current NC program is stopped after the current part program block has been completely processed. Otherwise, as for DB21, ... DBX7.3 (NC stop).
Signal state 0	No effect.
Corresponding to ...	DB21, ... DBX7.3 (NC stop) DB21, ... DBX7.4 (NC stop axes plus spindles) DB21, ... DBX35.2 (program status stopped) DB21, ... DBX35.6 (channel status interrupted)

DB21, ... DBX7.3	**NC stop**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	AUTOMATIC or MDI mode: Processing of the part program active in the channel is stopped. The axes (not spindles) are brought to a standstill within the assigned acceleration parameters. • Program status: Stopped • Channel status: Interrupted JOG mode: In JOG mode, incompletely traversed incremental paths (INC...) are retracted at the next NC start. Note The signal must be present for at least one PLC cycle (OB1).
Signal state 0	No effect.
Signal irrelevant for ...	• Program status: Aborted • Channel status: Reset
Special cases, errors, ...	• If data are transferred to the NCK after NC stop (e.g. tool offset), the data is cleared at the next NC start.
Corresponding to ...	DB21, ... DBX7.2 (NC stop at block limit) DB21, ... DBX7.4 (NC stop axes plus spindles) DB21, ... DBX35.2 (program status stopped) DB21, ... DBX35.6 (channel status interrupted)

DB21, ... DBX7.4	**NC stop axes plus spindles**
Edge evaluation: No	Signal(s) updated: Cyclically
	See DB21, ... DBX7.3 (NC stop). In addition, the spindles of the channel are stopped.

DB21, ... DBX7.7	Reset
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The channel is reset. The initial settings are set (e.g. G functions). The alarms for the channel are cleared if they are not POWER ON alarms. The reset signal must be issued by the PLC (e.g. using a logic operation with the reset key on the MCP). The signal is only evaluated by the selected channel. The program status changes to "Aborted", and the channel status changes to "Channel status reset".
Signal state 0	No effect.
Corresponding to ...	DB11, ... DBX0.7 (mode group reset) DB21, ... DBX35.7 (channel status reset)

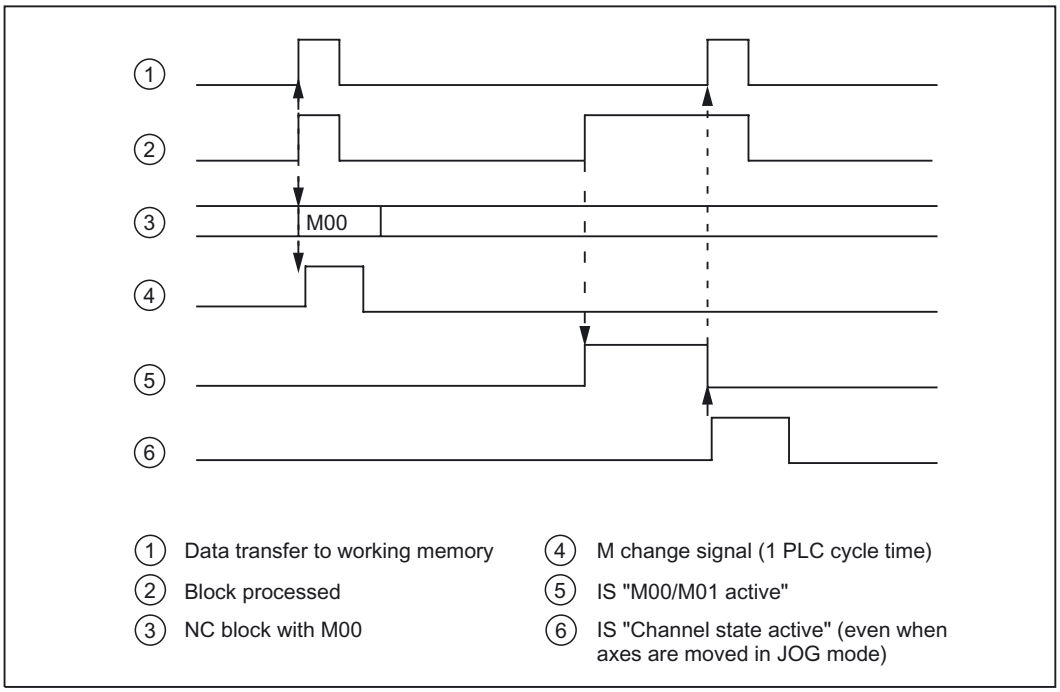
DB21, ... DBX31.0 - DBX31.2	REPOS mode (A, B, C)
Edge evaluation: No	Signal(s) updated: Cyclically
Corresponding to ...	DB21, ...DBX25.0-2 (REPOS mode) DB31, ... DBX10.0 (REPOS_DELAY)

DB21, ... DBX31.4	REPOS mode change
Edge evaluation: Yes	Signal(s) updated: Cyclically
Edge change 0 → 1	REPOS mode has changed.
Edge change 1 → 0	REPOS mode has not changed.
Corresponding to ...	DB21, ... DBX31.0-2 (REPOS mode) DB21, ... DBX319.0 (acknowledgement of REPOS mode change)

18.6.4 Signals from channel (DB21, ...)

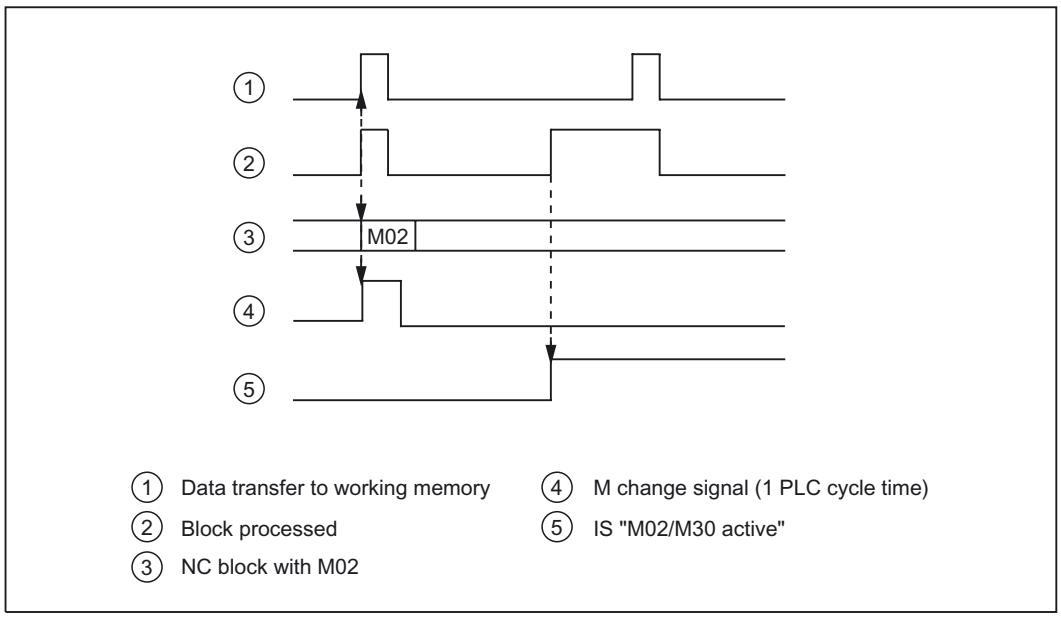
DB21, ... DBX32.3	Action block active	
Edge evaluation: No		Signal(s) updated: Cyclically
Signal state 1	The action block is being executed.	
Signal state 0	No action block active.	
Additional references	Operating Manual HMI (corresponding to the used software)	

DB21, ... DBX32.4	Approach block active	
Edge evaluation: No		Signal(s) updated: Cyclically
Signal state 1	The approach block for the progress of the program with "Block search with computation on contour" is active, as with "Block search with computation on block end point" no approach block is created of its own. The axes are automatically positioned on the collected search position if ASUP exits with REPOSA during "Block search with computation on contour".	
Signal state 0	The search target is found during "Block search with computation on contour".	
Further references	Programming Manual, Job Planning	

DB21, ... DBX32.5	**M00/M01 active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The part program block is processed, the auxiliary functions are output, and: • M00 is in the RAM • M01 is in the RAM and IS "Activate M01" is active The program status changes to "Stopped".
Signal state 0	• With DB21, ... DBX7.1 (NC start) • For a program abort as a result of a reset
Screen	
Corresponding to ...	DB21, ... DBX0.5 (activate `M01`) DB21, ... DBX24.5 (`M01` selected)

DB21, ... DBX32.6	Last action block active
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The last action block is being executed. This means, that all the action blocks on the side of NC have been processed and the actions on the side of PLC (ASUP, FC) or the operator such as overstore, mode change according to JOG/REPOS are possible. In this way the PLC for example can still perform a tool change before the start of movement.
Signal state 0	The last action block is not being executed. Action blocks contain the actions collected during "block search with computation" such as • Outputting help function H, M00, M01, M.. • Tool programming T, D, DL • Spindle programming S-Value, M3/M4/M5/M19, SPOS • Feed programming, F
Further references	See Section "K1: Mode group, channel, program operation, reset response (Page 489)"

DB21, ... DBX33.4	Block search active
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The block search function is active. It was selected from the operator interface screen and started using the interface signal: DB21, ... DBX7.1 (NC start)
Signal state 0	Search target found.
Application example(s)	The block search function makes it possible to jump to a certain block within a part program and to start processing the part program from this block.
Additional references	See Section "K1: Mode group, channel, program operation, reset response (Page 489)"

DB21, ... DBX33.5	**M02/M30 active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	• NC block with M02 or M30 (or M17 if a subprogram was started) is completely processed; if traversing motions are also programmed in this block, the signal is only output when the target position is reached. • The program was interrupted as a result of a reset, and the program status changes to "Aborted". • When MDI mode or machine functions REF or PRESET are selected • According to DB10 DBX56.2 (acknowledge emergency stop)
Signal state 0	• No program end or program abort • Status after activation of control • Start of an NC Program
Screen	
Application example(s)	The PLC can detect the end of program processing with this signal and react appropriately.
Special cases, errors, ...	• The M02 and M30 functions have equal priority. • The interface signal: DB21, ... DBX33.5 (M02/M30 active) is available as steady-state signal after the end of the program. • Not suitable for automatic follow-on functions such as workpiece counting, bar feed, etc. M02/M30 must be written in a separate block and the word M02/M30 or the decoded M signal used for these functions. • Auxiliary functions that could result in a read-in operation being stopped and any S values that are to be operative beyond M02/M30 must not be written in the last block of a program.

DB21, ... DBX33.6	**Transformation active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The NC command TRAORI (activate transformation) is programmed in the NC part program. This block has been processed by the NC and the transformation is now activated.
Signal state 0	No transformation active.
Additional references	Programming Manual, Job Planning

DB21, ... DBX33.7	**Program test active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	Program control program test is active. Axis disable is set internally for all axes (not spindles). Therefore the machine axes do not move when a part program block or a part program is being processed. The axis movements are simulated on the user interface with changing axis position values. The axis position values for the display are generated from the calculated setpoints. The part program is processed in the normal way.
Signal state 0	Program control "Program test" is not active.
Corresponding to ...	DB21, ... DBX1.7 (activate program test) DB21, ... DBX25.7 (program test selected)

DB21, ... DBX35.0	**Program status running**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The part program was started with the interface signal: DB21, ... DBX7.1 (NC start) and is running. The running program was stopped with the interface signal: DB21, ... DBX6.1 (read-in disable)
Signal state 0	• Program stopped by M00/M01 or NC stop or operating mode change. • If single block mode, the block is processed. • End of program reached (M02/M30). • Program aborted due to a reset. • No actual block in the memory (e.g. for MDI). • The actual block cannot be executed.
Signal irrelevant for ...	The part program was started with the interface signal: DB21, ... DBX7.1 (NC start) and is running.

DB21, ... DBX35.0	**Program status running**
Special cases, errors, ...	The interface signal: DB21, ... DBX35.0 (program status running) does not change to 0 if workpiece machining is stopped due to the following events: • A feed disable or spindle disable was output • DB21, ... DBX6.1 (read-in disable) • Feed correction to 0% • The spindle and axis monitoring functions respond • Position setpoints are entered in the NC program for axes in "follow-up mode," for axes without "servo enable," or for "parking axes"
Corresponding to ...	DB21, ... DBX6.1 (read-in disable)

DB21, ... DBX35.1	**Program status wait**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The running program is waiting for the program command `WAIT_M` or `WAIT_E` in an NC block. The wait condition specified in the `WAIT` command for the channel or channels has not yet been fulfilled.
Signal state 0	Program status wait is not active.
Further references	Programming Manual, Fundamentals

DB21, ... DBX35.2	**Program status stopped**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The NC part program has been stopped by: • DB21, ... DBX7.3 (NC stop) • DB21, ... DBX7.4 (NC stop axes plus spindles) • DB21, ... DBX7.2 (NC stop at the block limit) • Programmed `M00` or `M01` or • Single block mode
Signal state 0	"Program status stopped" is not present.
Corresponding to ...	DB21, ... DBX7.3 (NC stop) DB21, ... DBX7.4 (NC stop axes plus spindles) DB21, ... DBX7.2 (NC stop at the block limit)

DB21, ... DBX35.3	**Program status interrupted**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	When the operating mode changes from AUTOMATIC or MDI (in stopped program status) to JOG, the program status changes to "Interrupted". The program can be continued at the point of interruption in AUTOMATIC or MDI mode when NC start is issued.
Signal state 0	"Program status interrupted" not available.
Special cases, errors, ...	The interface signal: DB21, ... DBX35.3 (program status interrupted) indicates that the part program can continue to be processed by restarting it.

DB21, ... DBX35.4	**Program status aborted**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The program has been selected but not started, or the current program was aborted with a reset.
Signal state 0	Program status interrupted is not active.
Corresponding to ...	DB21, ... DBX7.7 (reset)

DB21, ... DBX35.5	**Channel status active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	In this channel: • A part program is presently being executed in Automatic or MDI mode or • At least one axis is being traversed in JOG mode.
Signal state 0	DB21, ... DBX35.3 (channel status interrupted) or DB21, ... DB35.7 (channel status reset) is present.

DB21, ... DBX35.6	**Channel status interrupted**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The NC part program in AUTOMATIC or MDI mode or a traversing motion in JOG mode can be interrupted by: • DB21, ... DBX7.3 (NC stop) • DB21, ... DBX7.4 (NC stop axes plus spindles) • DB21, ... DBX7.2 (NC stop at the block limit) • Programmed `M00` or `M01` or • Single block mode After an NC start the part program or the interrupted traversing movement can be continued.
Signal state 0	DB21, ... DBX35.5 (channel status active) or DB21, ... DB35.7 (channel status reset) is present.

DB21, ... DBX35.7	**Channel status reset**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The signal changes to 1 as soon as the channel goes into the reset state, i.e. no processing taking place.
Signal state 0	The signal is set to 0 as soon as processing takes place in the channel, e.g.: • Execution of a part program • Block search • TEACH IN active • Overstore active

DB21, ... DBX36.4	**Interrupt processing active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	One or more channels in the mode group are **not** in the desired operating mode as the result of an active interrupt routine. **Note:** The signal is not set if an interrupt routine is running in a program mode.
Signal state 0	All channels are operating in the required mode.
Corresponding to ...	MD11600 $MN_BAG_MASK

DB21, ... DBX36.5	Channel is ready
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	A channel is ready for a part program processing of machine axes, geometry axes and positioning axes. These are already allocated corresponding to machine configuration and the current program status of the concerned channels.
Signal state 0	The concerned channel is not ready for a part program processing of machine axes, geometry axes and positioning axes.
Corresponding to ...	MD11600 $MN_BAG_MASK

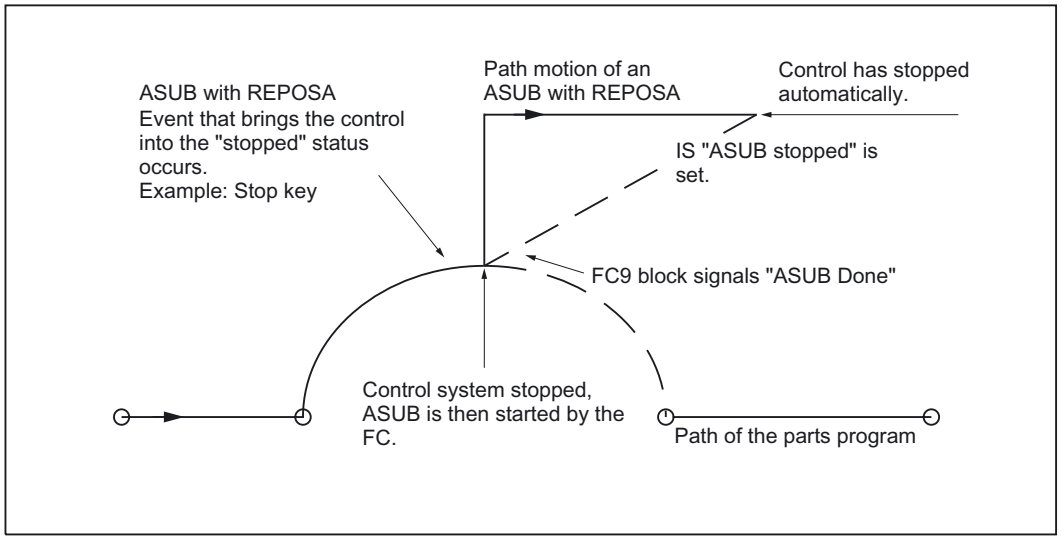
DB21, ... DBX37.6	Read-in disable is ignored
Edge evaluation: No	Signal(s) updated:
	The following machine data are used to specify that the read-in disable (DB21, ... DBX6.1) is to be ignored: • MD11602 $MN_ASUP_START_MASK, Bit 2 = 1 (start also permitted if read-in disable is active) • MD20116 $MC_IGNORE_INHIBIT_ASUP (execute interrupt program in spite of read-in disable) • MD20107 $MC_ PROG_EVENT_IGN_INHIBIT (Prog events ignore read-in disable) Part program blocks for which read-in disable is ignored are designated as read-in disable-inoperative.
Signal state 1	Read-in disable is active (DB21, ... DBX6.1==1) AND part program block is read-in disable-inoperative.
Signal state 0	Read-in disable is not active (DB21, ... DBB6.1 == 0) OR (read-in disable is active (DB21, ... DBX6.1 == 1) AND part program block is read-in disable-operative)
Corresponding to ...	DB21, ... DBX37.7 (Stop at block end is ignored during single block (SBL))

DB21, ... DBX37.7	Stop at block end is ignored during single block (SBL)
Edge evaluation:	Signal(s) updated:
	The following machine data and part program commands are used to specify that the stop at block end during single block (DB21, ... DBX0.4 == 1) is to be ignored: • MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop) • MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP (Execute interrupt program completely in spite of single block) • MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK (Prog events ignore single block) • SBLOF (suppress single block), SBLON (cancel single block suppression) Part program blocks for which stop at block end during single block is ignored are designated as single block-inoperative.
Signal state 1	Single block is active (DB21, ... DBX0.4==1) AND part program block is single block-inoperative.

DB21, ... DBX37.7	**Stop at block end is ignored during single block (SBL)**
Signal state 0	Single block is not active (DB21, ... DBB0.4 == 0) OR (single block is active (DB21, ... DBX0.4 == 1) AND part program block is single block-operative)
Corresponding to ...	Read-in disable is ignored. DB21, ... DBX37.6 (read-in disable is ignored)

DB21, ... DBB208 - DBB271	**Active G function of groups 1 to 60**								
Edge evaluation: No				Signal(s) updated: Cyclically					
Signal state <> 0	The G function displayed in BCD format or its mnemonic identifier is active in the G group.								
		Value	**Meaning**						
		1	1st G function of the G group						
		2	2nd G function of the G group						
		n	nth G function of the G group						
Signal state = 0	No G function or G group mnemonic identifier is active.								
Application example(s)	**Bit**	7	6	5	4	3	2	1	0
	Value	128	64	32	16	8	4	2	1
	Example:								
	G90	0	1	0	1	1	0	1	0
Special cases, errors, ...	In contrast to auxiliary functions, G functions are not output to the PLC subject to acknowledgement, i.e. processing of the part program is continued immediately after the G function output.								
Additional references	A complete list of the G groups and G functions can be found in: **References**: Programming Manual, Fundamentals.								

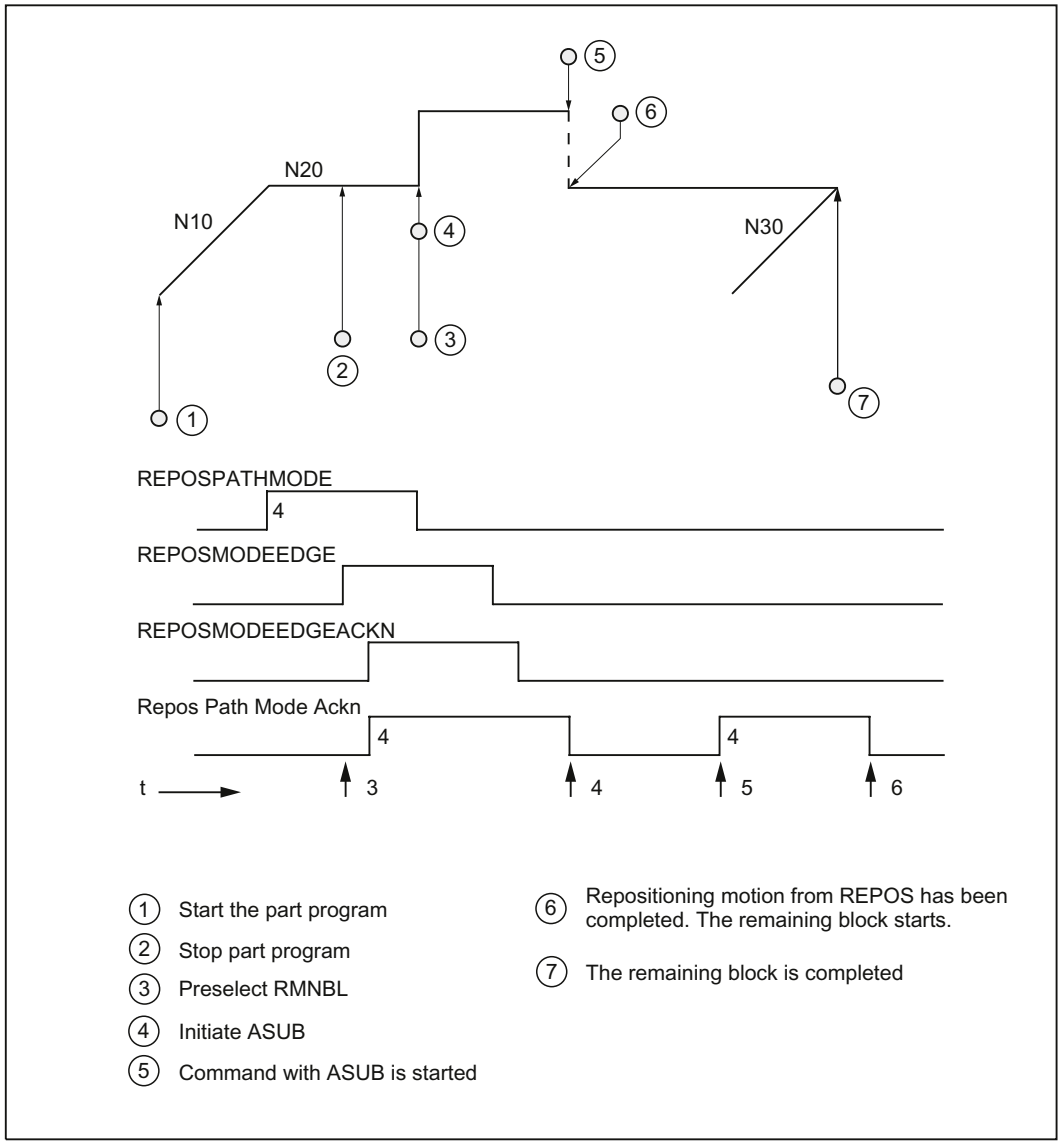
DB21, ... DBB317.1	**Workpiece setpoint reached**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state = 1	The number of machined workpieces (actual workpiece total) is **equal** to the number of workpieces to be machined (required number of workpieces): $AC_ACTUAL_PARTS == $AC_REQUIRED_PARTS
Signal state = 0	The number of machined workpieces (actual workpiece total) is **not equal** to the number of workpieces to be machined (required number of workpieces): $AC_ACTUAL_PARTS <> $AC_REQUIRED_PARTS
Corresponding to ...	MD27880 $MC_PART_COUNTER (activation of workpiece counters)
Additional references	Function Manual, Fundamentals, Section "K1: Mode group, channel, program operation, reset response" > "Program runtime / workpiece counter" > "Workpiece counter"

DB21, ... DBX318.0	**ASUB is stopped**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The signal is set to 1 if the control stops automatically prior to the end of the ASUB. The IS DB21, ... DBX318.0 (ASUB is stopped) is only supplied in the case "Interrupt in a program mode and channel status stopped".
Signal state 0	The IS DB21, ... DBX318.0 (ASUB is stopped) is set to 0 with start and reset.
	Typical sequence of an ASUB with REPOSA: 
ASUB with REPOSA is triggered in the status AUTOMATIC mode stopped	If the interrupt routine is ended with REPOSA, then the following sequence is typical: • The part program is stopped using the stop key, stop-all key or as a result of an alarm. • The control assumes program status "Stopped". • The PLC initiates an ASUB via block FC9. • Before the re-approach to the contour, the control stops and goes to program state "Stopped". IS DB21, ... DBX318.0 (ASUB is stopped) is set. • The user presses Start. The IS DB21, ... DBX318.0 (ASUB is stopped) is reset, the re-approach motion is started. • At the end of the re-approach motion, the FC9 signal "ASUB done" is set and the path of the interrupted part program is continued.

DB21, ... DBX318.1	**Block search via program test is active (SERUPRO)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	During the processing of the part program block in context of block search (internal channel status: "Program test"), up to the change of the target block in the main execution (Program status: "Stopped").
Signal state 0	With the change of the target block in the main execution (internal channel status: "Program test" is deselected; stop condition: "Search target found" is displayed).
Special cases, errors, ...	The block search (SERUPRO) can only be activated in AUTOMATIC mode in program status "Aborted".

DB21, ... DBX319.0	**Acknowledgement of the REPOS mode change**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Edge change 0 → 1	The interface signal detected by the NC: DB21, ... DBX31.4 (REPOS mode change) is acknowledged with the existing interface signal, if the requested REPOS mode: DB21, ... DBX31.0-2 (REPOS mode) and the delay signal: DB31, ... DBX10.0 (REPOSDELAY) are taken over in the NC. The signal states refer to the current main run block
Edge change 1 → 0	SERUPRO-ASUB stops automatically before REPOS and DB21, ... DBX31.4 (REPOS mode change) does not affect the SERUPRO approach.
Corresponding to ...	DB21, ... DBX31.4 (REPOS mode-change)

DB21, ... DBX319.1 - DBX319.3	**Active REPOS mode (A, B, C)**			
Edge evaluation: No	Signal(s) updated: Cyclically			
Meaning	The active REPOS mode is displayed with the interface signals A, B and C.			
	Active REPOS mode	**C**	**B**	**A**
	0 = no REPOS mode active	0	0	0
	1 = repositioning to block start point `RMBBL`	0	0	1
	2 = repositioning to interruption point `RMIBL`	0	1	0
	3 = repositioning to block end point `RMEBL`	0	1	1
	4 = repositioning to nearest path point `RMNBL`	1	0	0

DB21, ... DBX319.1 - DBX319.3	**Active REPOS mode (A, B, C)**
	Example of the sequence of REPOS acknowledgements in the part program and signal timing of the acknowledgement process from the NC:  ① Start the part program ② Stop part program ③ Preselect RMNBL ④ Initiate ASUB ⑤ Command with ASUB is started ⑥ Repositioning motion from REPOS has been completed. The remaining block starts. ⑦ The remaining block is completed
Corresponding to ...	DB21, ... DBX31.0-2 (REPOS mode) DB21, ... DBX31.4 (REPOS mode-change) DB21, ... DBX319.0 (acknowledgement of REPOS mode change) DB31, ... DBX70.2 (acknowledgement of REPOS delay)
Additional references	See Section "Block search Type 5 SERUPRO (Page 537)"

DB21, ... DBX319.5	Repos DEFERAL Chan
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	All axes currently controlled by this channel have either no REPOS offset or their REPOS offsets are invalid.
Signal state 0	Miscellaneous.
Corresponding to ...	DB31, ... DBX70.0 (Repos offset)

DB21, ... DBB376	ProgEventDisplay	
Edge evaluation: No	Signal(s) updated: Event-driven **Note:** Signal duration of at least one complete PLC cycle	
	All bits == 0: There is no event-driven program call active Meaning of individual bits:	
	Bit	**Meaning**
	0	Part program start from channel state "Reset"
	1	Part program end
	2	Operator panel reset
	3	Startup
	4	1st start after block search
	5 - 7	Reserved, currently always 0
Signal state 1	The event assigned to the bit has activated the "Event-driven program call" function.	
Signal state 0	The event assigned to the bit has not activated the "Event-driven program call" function or the event-driven user program has expired or was canceled with RESET.	

DB21, ... DBX378.0	ASUB is active
Edge evaluation: No	Signal(s) updated: Event-driven
Signal state 1	One ASUB is active. **Note:** The user gets a feedback on a running ASUB through DB21, … DBX378.0 even outside FC9 block.
Signal state 0	No ASUB is active.

DB21, ... DBX378.1	**Still ASUB is active**
Edge evaluation: No	Signal(s) updated: Event-driven
Signal state 1	An ASUB with suppressed display updating (refer to MD20191 $MC_IGN_PROG_STATE_ASUP) is active.
Signal state 0	No ASUB with suppressed display updating is active.

DB21, ... DBX384.0	**Control program branching**
Edge evaluation: No	Signal(s) updated:
Signal state 1	GOTOS in the part program initiates a return to the program start. The program is then processed again.
Signal state 0	GOTOS initiates no return. Program execution is continued with the next part program block after GOTOS.
Corresponding to ...	MD27860 $MC_PROCESSTIMER_MODE (Activation of the program runtime measurement) MD27880 $MC_PART_COUNTER (activation of workpiece counters)

18.6.5 Signals to axis/spindle (DB31, ...)

DB31, ... DBX10.0	**REPOSDELAY**
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The REPOS offset of the axis is first applied with its next programming.
Signal state 0 or edge change 1 → 0	There is no REPOS offset.
Special cases, errors, ...	The signal is not relevant for path axes.
Corresponds to	DB21, ... DBX31.0 - DBX31.2 (REPOSPATHMODE0-2) DB31, ... DBX70.2 (REPOS Delay Ackn) DB31, ... DBX72.0 (REPOSDELAY)

18.6.6 Signals from axis/spindle (DB31, ...)

DB31, ... DBX70.0	REPOS offset
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	A REPOS offset must be applied for the appropriate axis.
Signal state 0 or edge change 1 → 0	No REPOS offset need be applied for the appropriate axis.
Corresponds to	DB31, ... DBX70.1 (REPOS offset valid)

DB31, ... DBX70.1	REPOS offset valid
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The range of validity of the REPOS offset is indicated with the value 1. The REPOS offset was calculated to be valid.
Signal state 0 or edge change 1 → 0	A value of zero indicates that the REPOS offset was calculated to be invalid.
Application example(s)	Updating the REPOS offset in the range of validity: Between SERUPRO end and start, the axis can be moved in JOG mode with a mode change. The user moves the REPOS offset to the zero value.
Corresponds to	DB31, ... DBX70.0 (REPOS offset)

DB31, ... DBX70.2	REPOS Delay Ackn
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The axis was programmed within a traversing block, and the REPOS offset was applied. Note A REPOS offset was available for the axis, and REPOSDELAY was active: DB31, ... DBX10.0 (REPOSDELAY) == 1 This signal behaves the same as: DB21, ... DBX319.1 - DBX319.3 (Repos Path Mode Ackn)
Signal state 0 or edge change 1 → 0	The value zero is used to acknowledge that the REPOS offset is not active for this axis. This signal is cancelled on activation of the remaining block.
Corresponds to	DB31, ... DBX10.0 (REPOSDELAY) DB31, ... DBX72.0 (REPOSDELAY)

DB31, ... DBX72.0	REPOSDELAY
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	After a block search, a REPOS offset is applied for this axis. However it is not applied using the approach block, but rather using the next traversing block in which the axis is programmed.
Signal state 0 or edge change 1 → 0	The REPOS offset for this axis is not active.
Special cases, errors, ...	The signal is not relevant for path axes.
Corresponds to	DB21, ... DBX31.0 - DBX31.2 (REPOSPATHMODE) DB31, ... DBX10.0 (REPOSDELAY) DB31, ... DBX70.2 (REPOS Delay Ackn)

DB31, ... DBX76.4	Path axis
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The axis is involved in the path (path axis). Note In conjunction with SERUPRO in status "Target block found", the signal refers to the status of the axis in the target block.
Signal state 0 or edge change 1 → 0	The axis is not involved in the path.

18.7 Axis types, coordinate systems, frames (K2)

18.7.1 Signals to axis/spindle (DB31, ...)

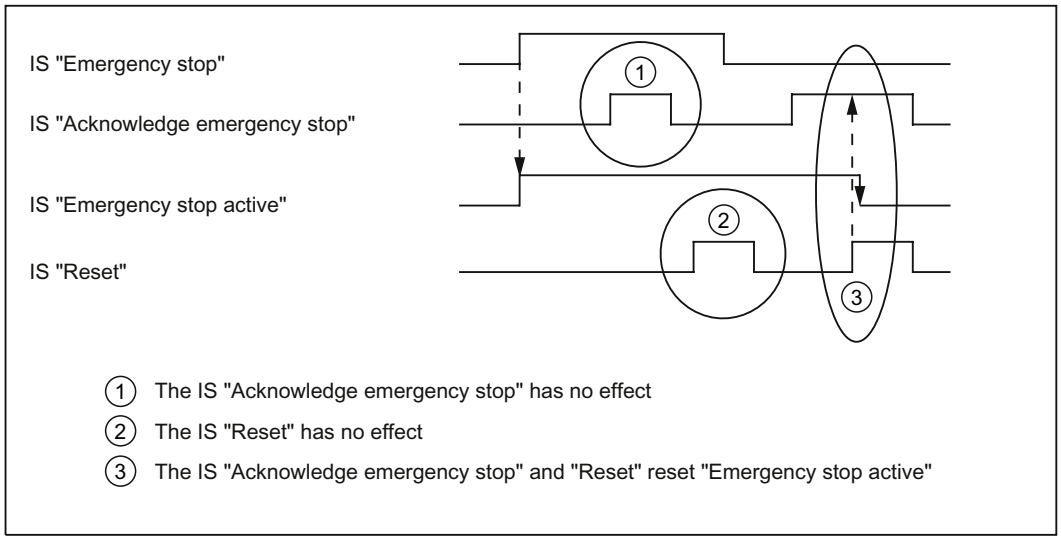
DB 31, ... DBX3.0	**External Zero Offset**
Edge evaluation: no	Signal(s) updated:
Signal state 1 or edge change 0 → 1	The preselected value of the external work offest of an axis is used as the new value for calculating the total work offset between the basic and the workpiece coordinate systems.
Signal state 0 or edge change 1 → 0	The preselected value of the external work offset of an axis is not used as the new value for calculating the total work offset between the basic and workpiece coordinate systems. The previous value is still valid.
Signal irrelevant for ...	$AA_ETRANS[axis] equals zero for all axes.
Special cases, errors, ...	Signal zero after ramp-up (power ON).
Corresponding to	$AA_ETRANS[axis]

18.8 Emergency stop (N2)

18.8.1 Signals to NC (DB10)

DB10 DBX56.1	Emergency stop
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The NC is brought into the emergency stop state and the emergency stop sequence in the NC is started.
Signal state 0 or edge change 1 → 0	The NC is not in the emergency stop state. The emergency stop state is (still) active but can be reset using the interface signals: DB10 DBX56.2 (acknowledge emergency stop) and DB11 DBX0.7 (mode group reset)
Corresponding to ...	DB10 DBX56.2 (acknowledge emergency stop) DB10 DBX106.1 (emergency stop active)

DB10 DBX56.2	Acknowledge emergency stop
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The emergency stop state is only reset if the interface signal: DB10 DBX56.2 (acknowledge emergency stop) is set followed by the interface signal: DB11, ... DBX0.7 (mode group reset). It must be noted that IS "Acknowledge emergency stop" and IS "Reset" must be set (together) for a long enough period so that the interface signal: DB10 DBX106.1 (emergency stop active) was reset. Resetting the emergency stop state has the following effects: • The controller enable is switched in. • Follow-up mode is canceled for all axes and position control mode resumed. • DB31, ... DBX61.5 set (position controller active). • DB11, ... DBX6.3 set (mode group ready). • DB10 DBX106.1 reset (emergency stop active). • Alarm 3000 is cleared. • Part program processing is aborted for all channels.

DB10 DBX56.2	**Acknowledge emergency stop**
	
Special cases, errors, ...	The emergency stop state cannot be reset using the interface signal: DB21, ... DBX7.7 (reset).
Corresponding to ...	DB10 DBX56.1 (Emergency stop) DB10 DBX106.1 (emergency stop active) DB11 DBX0.7 (mode group reset)

18.8.2 Signals from NC (DB10)

DB10 DBX106.1	**Emergency stop active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The NC is in the emergency stop state.
Corresponding to ...	DB10 DBX56.1 (Emergency stop) DB10 DBX56.2 (acknowledge emergency stop)

18.9 PLC basic program (P3)

To describe the NC/PLC interface signals, refer to:
References:
Functions Manual, Basic Functions; PLC Basic Program (P3)
Chapter: "Signal/Data Specifications"

18.10 Reference point approach (R1)

18.10.1 Signals to channel (DB21, ...)

DB21, ... DBX1.0	**Activate referencing**	
Edge evaluation: Yes		Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Channel-specific referencing is started using the interface signal: DB21, ... DBX1.0 (activate referencing). The control acknowledges a successful start using the interface signal: DB21, ... DBX33.0 (referencing active) With the channel-specific referencing each machine axis, which is allocated to a channel, can be referenced (control internals are simulated by traversing keys plus/minus). Using the axis-specific machine data: MD34110 $MA_REFP_CYCLE_NR (axis sequence for channel-specific referencing) can determine in which sequence the machine axes are referenced. When all of the axes entered in REFP_CYCLE_NO have reached their reference point, the interface signal: DB21, ... DBX36.3 (all axes stationary) is set.	
Application example(s)	If the machine axes are to be referenced in a particular sequence, there are the following possibilities: • The operator must observe the correct sequence when starting. • The PLC must check the sequence when starting or define it itself. • The channel-specific referencing function will be used.	
Corresponding to ...	DB21, ... DBX33.0 (activate referencing) DB21, ... DBX36.2 (all axes with obligatory reference point are referenced)	

DB21, ... DBX33.0	Referencing active
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The channel-specific referencing was started using the interface signal: DB21, ... DBX1.0 (activate referencing) and the successful start was acknowledged using the interface signal: DB21, ... DBX33.0 (referencing active). The channel-specific referencing is operational.
Signal state 0 or edge change 1 → 0	• Channel-specific referencing is completed • Axis-specific referencing is running • No referencing active
Signal irrelevant for ...	Spindles
Corresponding to ...	DB21, ... DBX1.0 (activate referencing)

18.10.2 Signals from channel (DB21, ...)

DB21, ... DBX36.2	All axes that have to be referenced are referenced
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	All axes that must be referenced (linear axes and rotary axes) of the channel are referenced. The machine data: MD20700 $MC_REFP_NC_START_LOCK (NC start inhibit without reference point) is zero. If two position measuring systems are connected to an axis, that would prevent an NC start, then the active one must be referenced so that the axis is considered to have been referenced. An NC Start command for parts program processing is only accepted when this signal is present. Axes that have to be referenced are axes, if: MD34110 $MA_REFP_CYCLE_NR _ = -1 and the axis is not in the parked position (position measuring system inactive and the controller enable withdrawn).
Signal state 0 or edge change 1 → 0	One or more axes of the channel have not been referenced.
Special cases, errors, ...	The spindles of the channel have no effect on this interface signal.
Corresponding to	DB31, ... DBX60.4 (referenced / synchronized 1) DB31, ... DBX60.5 (referenced / synchronized 2)

18.10.3 Signals to axis/spindle (DB31, ...)

DB31, ... DBX2.4 - DBX2.7	Reference point value 1 to 4
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	When the reference cam is reached, the NCK is signaled which coded reference cam is actuated. The interface signal: DB31, ... DBX2.4 - DBX2.7 (reference point values 1 to 4) must remain set until the reference point is reached, or until a new coded reference cam is actuated. If the machine axis has reached the reference point (axis stationary) then using the reference point value pre-selected using IS "reference point values 1 to 4" from the machine data: MD34100 $MA_REFP_SET_POS (reference point value) is transferred into the control as the new reference position.
Signal state 0 or edge change 1 → 0	No effect.
Signal irrelevant for ...	Linear measurement systems with distancecoded reference marks
Application example(s)	On a machine tool with large traversing distances, four coded reference cams can be distributed over the entire distance traveled by the axis, four different reference points approached and the time required to reach a valid referenced point reduced.
Special cases, errors, ...	If the machine axis has arrived at the reference point and none of the four "reference point value 1 to 4" interface signals has been set, the value of the reference point is automatically set to 1.
Corresponding to	MD34100 $MA_REFP_SET_POS (reference point value)

DB31, ... DBX12.7	Reference point approach delay
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The machine axis is positioned on the reference cam.
Signal state 0 or edge change 1 → 0	The machine axis is positioned in front of the reference cam. An appropriately long reference cam (up to the end of the traversing range) should be used to prevent the machine axis from being located behind (after) the referencing cam.
Corresponding to	DB31, ... DBX2.4 - DBX2.7 (reference point values 1 to 4)

18.10.4 Signals from axis/spindle (DB31, ...)

DB31, ... DBX60.4	Referenced/synchronized 1
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	**Axes:** When being referenced, if the machine axis has reached the reference point (incremental measuring systems) or the target point (for length measuring system with distance-coded reference marks), then the machine axis is referenced and the following interface signal is set: DB31, ... DBX60.4 (referenced/synchronized 1) (depending on which position measuring system is active when referencing). **Spindles:** After power-on, a spindle is synchronized the latest after one spindle revolution (360 degrees) (the zero mark passed or the Bero responded).
Signal state 0 or edge change 1 → 0	The machine axis/spindle with position measuring system 1 is not referenced/synchronized.
Corresponding to ...	DB31, ... DBX1.5 (position measuring system 1)

DB31, ... DBX60.5	Referenced/synchronized 2
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	**Axes:** When being referenced, if the machine axis has reached the reference point (incremental measuring systems) or the target point (for length measuring system with distance-coded reference marks), then the machine axis is referenced and the following interface signal is set: DB31, ... DBX60.5 (referenced/synchronized 2) (depending on which position measuring system is active when referencing). **Spindles:** After power-on, a spindle is synchronized the latest after one spindle revolution (360 degrees) (the zero mark passed or the Bero responded).
Signal state 0 or edge change 1 → 0	The machine axis/spindle with position measuring system 2 is not referenced/synchronized. **Axes:** Alarm 21610 was output. **Spindles:** Encoder limit frequency exceeded.
Corresponding to ...	DB31, ... DBX1.6 (position measuring system 2) MD34102 $MA_REFP_SYNC_ENCS (measuring system calibration) = 0

DB31, ... DBX71.4	POS_RESTORED 1
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	If MD34210 $MA_ENC_REFP_STATE is set to a value of 3, the last axis position buffered before switch off is restored in distance-coded, incremental measuring systems. Referencing does not take place automatically. Position measuring system 1 is in "Position restored" state.
Signal state 0 or edge change 1 → 0	The machine axis/spindle with position measuring system 1 is not restored.
Corresponding to ...	DB31, ... DBX60.4 (referenced / synchronized 1)

DB31, ... DBX71.5	POS_RESTORED 2
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	If MD34210 $MA_ENC_REFP_STATE is set to a value of 3, the last axis position buffered before switch off is restored in distance-coded, incremental measuring systems. Referencing does not take place automatically. Position measuring system 2 is in "Position restored" state.
Signal state 0 or edge change 1 → 0	The machine axis/spindle with position measuring system 2 is not restored.
Corresponding to ...	DB31, ... DBX60.5 (referenced / synchronized 2)

18.11 Spindles (S1)

18.11.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX2.2	Spindle reset/delete distancetogo
Edge evaluation: yes	Signal(s) updated: cyclic
Edge change 0 → 1	Independent of the machine data: MD35040 $MA_SPIND_ACTIVE_AFTER_RESET selects a spindle reset for the various spindle operating modes in the following fashion: • Control mode: – Spindle stops – Program continues to run – Spindle continues to run with subsequent M and S program commands • Oscillating mode: – Oscillation is interrupted – Axes continue to run – Program continues with the actual gearbox stage – With the following M value and higher S value, it is possible that the IS: DB31, ... DBX83.1 (programmed speed high) is set. • Positioning mode: – Is stopped • Axis operation: – Is stopped
Signal state 0 or edge change 1 → 0	No effect.
Corresponding to ...	MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (own spindle reset) DB21, ... DBX7.7 (reset) DB31, ... DBX2.2 (delete distance to go): is a different name for the same signal

DB31, ... DBX16.0 - DBX16.2	Actual gear stage A to C
Edge evaluation: yes	Signal(s) updated: Cyclic
Signal state 1 (statuscontrolled)	If the new gear stage is engaged, the PLC user program sets the interface signals: DB31, ... DBX16.2 - DBX16.0 (actual gear stage A to C) and DB31, ... DBX16.3 (gear is changed over). This signals the NCK that the correct gear stage has been successfully engaged. The gear change is considered to have been completed (spindle oscillation mode is deselected), the spindle accelerates in the new gear stage to the last programmed spindle speed and the next block in the parts program can be executed. The actual gear stage is output in coded format. For each of the 5 gear stages, there is one set of parameters assigned as follows:

Parameter set No.	NC/PLC interface	Meaning
0	–	Data for axis mode
1	000	Data for 1st gear stage
	001	
2	010	Data for 2nd gear stage
3	011	Data for 3rd gear stage
4	100	Data for 4th gear stage
5	101	Data for 5th gear stage
	110	
	111	

Special cases, errors, ...	If the PLC user reports back to the NCK with a different actual gear stage than issued by the NCK as the setpoint gear stage, the gear change is still considered to have been successfully completed and the actual gear stage A to C is activated.
Corresponding to ...	DB31, ... DBX82.0 - DBX82.2 (setpoint gear stage A to C) DB31, ... DBX82.3 (change over gear stage) DB31, ... DBX16.3 (gear is changed) DB31, ... DBX18.5 (oscillation speed) Parameter sets for gear stages

DB31, ... DBX16.3	Gear is changed over
Edge evaluation: yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	When the new gear stage is engaged, the PLC user sets the interface signals: DB31, ... DBX16.0 - DBX16.2 (actual gear stages A to C) and DB31, ... DBX16.3 (gear is changed over). This signals the NCK that the correct gear stage has been successfully engaged. The gear stage change is complete (spindle oscillation mode is deselected), the spindle accelerates in the new gear stage to the last programmed spindle speed and the next block in the parts program can be executed. The interface signal: DB31, ... DBX82.3 (change over gear) is reset by the NCK - the PLC user then resets the interface signal: (gear is changed over).
Signal state 0 or edge change 1 → 0	No effect.
Signal irrelevant for ...	... spindle modes other than the oscillation mode
Special cases, errors, ...	If the PLC user reports back to the NCK with a different actual gear stage than issued by the NCK as the setpoint gear stage, the gear change is still considered to have been successfully completed and the actual gear stage A to C is activated.
Corresponding to ...	DB31, ... DBX16.2 - DBX16.0 (actual gear stage A to C) DB31, ... DBX82.2 - DBX82.0 (setpoint gear stage A to C) DB31, ... DBX82.3 (change over gear stage) DB31, ... DBX18.5 (oscillation speed)

DB31, ... DBX16.4 - DBX16.5	**Resynchronizing spindles 1 and 2**
Edge evaluation: yes	Signal(s) updated: Cyclic
Edge change 0 → 1	The spindle should be resynchronized, as the synchronization between the position measuring system of the spindle and the 0 degree position has been lost.
Signal state 0 or edge change 1 → 0	No effect.
Signal irrelevant for ...	... spindle modes other than the control mode.
Application example(s)	The machine has a selector switch to changeover between a vertical and a horizontal spindle. Two different position encoders are used (one for the vertical spindle and one for the horizontal spindle), but only one actual value input is used on the control. When the system switches from the vertical to the horizontal spindle, the spindle must be resynchronized. This synchronization is triggered by the IS "re-synchronize spindle 1 or 2".
Corresponding to ...	DB31, ... DBX60.4 (referenced / synchronized 1) DB31, ... DBX60.5 (referenced / synchronized 2)

DB31, ... DBX16.7	**Delete S value**
Edge evaluation: yes	Signal(s) updated: Cyclic
Edge change 0 → 1	**Control mode:** • Spindle stops • Program continues to run • Spindle continues to run with the following S value, if `M3` or `M4` were active **Oscillation mode, axis mode, positioning mode:** • Signal has no effect for the corresponding function. However, if the open-loop control mode is selected again, a new S value must be programmed.
Signal state 0 or edge change 1 → 0	No effect.
Application example(s)	Terminating traversing motion on account of an external signal (e.g. sensing probe).

DB31, ... DBX17.4 - DBX17.5	**Re-synchronizing spindle when positioning 1 and 2**
Edge evaluation: yes	Signal(s) updated: Cyclic
Signal state 1	When positioning, the spindle must be re-synchronized.
Signal state 0 or edge change 1 → 0	No effect.
Signal irrelevant for ...	... spindle modes other than the positioning mode.
Application example(s)	The spindle has an indirect measuring system and slippage may occur between the motor and the clamp. If the signal=1 - when positioning is started, the old reference is deleted and the zero mark is searched for again before the end position is approached.
Corresponding to ...	DB31, ... DBX60.4 (referenced / synchronized 1) DB31, ... DBX60.5 (referenced / synchronized 2)

DB31, ... DBX17.6	**Invert M3/M4**
Edge evaluation: yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The direction of rotation of the spindle motor changes for the following functions: • M3 • M4 • M5 • SPOS/M19/SPOSA from the motion; not effective for SPOS/M19/SPOSA from zero speed (stationary).
Application example(s)	The machine has a selector switch to changeover between a vertical and a horizontal spindle. The mechanical design is implemented so that for the horizontal spindle, one more gearwheel is engaged than for the vertical spindle. The direction of rotation must therefore be changed for the vertical spindle if the spindle is always to rotate clockwise with M3.

DB31, ... DBX18.4	Oscillation controlled by the PLC
Edge evaluation: yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	If the interface signal: DB31, ... DBX18.4 (oscillation controlled by thePLC) is not set, then automatic oscillation in the NCK is carried-out using the interface signal: DB31, ... DBX18.5 (oscillation speed). The two times for the directions of rotation are entered in the machine data: MD35440 $MA_SPIND_OSCILL_TIME_CW (oscillation time for the M3 direction) and MD35450 $MA_SPIND_OSCILL_TIME_CCW (oscillation time for the M4 direction). If the IS "oscillation via PLC" is set, then with the IS "oscillation speed" a speed is only output in conjunction with the interface signals: DB31, ... DBX18.6 - DBX18.7 (setpoint direction of rotation, counter-clockwise and clockwise). The oscillation, i.e. the continuous change of the direction of rotation, is performed by the PLC user program using the interface signal "setpoint direction of rotation, counter-clockwise and clockwise" (oscillation via the PLC).
Application example(s)	If the new gear stage cannot be engaged in spite of several attempts by the NCK, the system can be switched to oscillation via PLC. Both of the times for the directions of rotation can then be altered by the PLC user program as required. This ensures that the gear stage is reliably changed - even with unfavorable gear wheel positions.
Corresponding to ...	MD35440 $MA_SPIND_OSCILL_TIME_CW (oscillation time for the M3 direction) MD35450 $MA_SPIND_OSCILL_TIME_CCW (oscillation time for the M4 direction) DB31, ... DBX18.5 (oscillation speed) DB31, ... DBX18.7 (setpoint direction of rotation, counter-clockwise) DB31, ... DBX18.6 (setpoint direction of rotation, clockwise)

DB31, ... DBX18.5	Oscillation enable
Edge evaluation: no	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	If the gear stage is to be changed (DB31, ... DBX82.3 (change over gear) is set), then the spindle operating mode changes to the oscillation mode. Depending on the instant in time that the interface signal: DB31, ... DBX18.5 (oscillation speed) is set, the spindle decelerates down to standstill with different deceleration levels: The IS "Oscillation speed" is set before the interface signal: DB31, ... DBX82.3 (change over gear) is set by the NCK. When oscillating, the spindle is decelerated down to standstill with the deceleration: MD35410 $MA_SPIND_OSCILL_ACCEL. Once the spindle is stationary, oscillation is immediately initiated. The IS "Oscillation speed" is enabled after the IS "Change gear" is set by the NCK and when the spindle is stationary. The position controller is disabled. The spindle decelerates with the specified deceleration rate in the speed controlled mode. After the IS "oscillation speed" is set, the spindle starts to oscillate with the oscillation acceleration (MD35410). if the interface signal: DB31, ... DBX18.4 (oscillation via the PLC) is not set, then automatic oscillation is executed in the NCK using the IS "Oscillation speed". The two times for the directions of rotation are entered in the machine data: MD35440 $MA_SPIND_OSCILL_TIME_CW (oscillation time for the M3 direction) and MD35450 $MA_SPIND_OSCILL_TIME_CCW (oscillation time for the M4 direction). If the IS "oscillation via PLC" is set, then with the IS "oscillation speed" a speed is only output in conjunction with the interface signals: DB31, ... DBX18.6 - DBX18.7 (setpoint direction of rotation, counter-clockwise and clockwise). The oscillation, i.e. the continuous change of the direction of rotation, is performed by the PLC user program using the interface signal "setpoint direction of rotation, counter-clockwise and clockwise" (oscillation via the PLC).
Signal state 0 or edge change 1 → 0	The spindle does not oscillate.
Signal irrelevant for ...	... All spindle modes except the oscillation mode.
Application example(s)	The oscillation speed is used to make it easier to engage a new gear stage.
Corresponding to ...	DB31, ... DBX18.4 (oscillation controlled by the PLC) DB31, ... DBX18.7 (setpoint direction of rotation, counter-clockwise) DB31, ... DBX18.6 (setpoint direction of rotation, clockwise)

DB31, ... DBX18.6 - DBX18.7	Oscillation direction of rotation counter-clockwise / oscillation direction of rotation clockwise
Edge evaluation: yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	If the interface signal: DB31, ... DBX18.4 (oscillation by the PLC) is set, then the direction of rotation for the oscillation speed can be entered using the two interface signals: DB31, ... DBX18.6 - DBX18.7 (setpoint direction of rotation counter-clockwise and clockwise). The times for the oscillation movement of the spindle motor are defined by setting the interface signals "direction of rotation setpoint counter-clockwise and clockwise" for a corresponding length of time.
Signal irrelevant for ...	... spindle modes other than the oscillation mode
Application example(s)	Refer to DB31, ... DBX18.4 (oscillation controlled by the PLC)
Special cases, errors, ...	If both of the interface signals are set simultaneously enabled, no oscillation speed is output. If an interface signal is not set, then an oscillation speed is not output.
Corresponding to ...	DB31, ... DBX18.4 (oscillation controlled by the PLC DB31, ... DBX18.5 (oscillation speed)

18.11.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX60.0	Spindle/no axis
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The machine axis is operated in one of the following spindle modes: • Control mode • Oscillation mode • Positioning mode • Tapping without compensating chuck • Synchronous mode The interface signals to the axis (DB31, ... DBB12 - DBB15) and from the axis (DB31, ... DBB74 - DBB81) are **invalid**. The interface signals to the spindle (DB31, ... DBB16 - DBB19) and from the spindle (DB31, ... DBB82 - DBB91) are **valid**.

DB31, ... DBX60.0	**Spindle/no axis**
Signal state 0 or edge change 1 → 0	The machine axis is operated as an axis The interface signals to the axis (DB31, ... DBB12 - DBB15) and from the axis (DB31, ... DBB74 - DBB81) are **valid**. The interface signals to the spindle (DB31, ... DBB16 - DBB19) and from the spindle (DB31, ... DBB82 - DBB91) are **invalid**.
Application example(s)	If a machine axis operates alternatively as a spindle or rotary axis: • Turning machine: Spindle / C axis • Milling machine: Spindle / rotary axis for rigid tapping The currently active operating mode can be identified from interface signal: DB31, ... DBX60.0 (spindle/no axis).

DB31, ... DBX82.0 - 82.2	**Setpoint gear stage A to C**
Edge evaluation: Yes	Signal(s) updated: Cyclically
	See DB31, ... DB82.3 (change gear).
Signal state 1 or edge change 0 → 1	The set gear stage is output in coded format: 1st gear stage 0 0 0 (C B A) 1st gear stage 0 0 1 2nd gear stage 0 1 0 3rd gear stage 0 1 1 4th gear stage 1 0 0 5th gear stage 1 0 1 invalid value 1 1 0 invalid value 1 1 1
Signal irrelevant for ...	... Other spindle modes except oscillation mode
Corresponding to ...	DB31, ... DBX82.3 (change gear) DB31, ... DBX16.0 - DBX16.2 (actual gear stage A to C) DB31, ... DBX16.3 (gear is changed)

DB31, ... DBX82.3	**Change gear**
Edge evaluation: Yes	Signal(s) updated: Cyclically
	Specification of gear stage: • Manual specification using M function M41 - M45 corresponding to gear stage 1 - 5 If set gear stage <> actual gear stage => DB31, ... DBX82.3 (change gear) = 1 DB31, ... DBX82.0 - DBX82.2 (set gear stage) = set gear stage • Automatic gear stage selection depending on the progr. Spindle speed via M-function M40 at specified setpoint speed requires gear stage change => DB31, ... DBX82.3 (change gear) = 1 DB31, ... DBX82.0 - DBX82.2 (setpoint gear stage) = setpoint gear stage
Signal state 1 or edge change 0 → 1	New set gear stage was specified AND set gear stage <> actual gear stage
Special cases, errors, ...	Signal is not output if: Set gear stage == actual gear stage
Corresponding to ...	DB31, ... DBX82.0 - DBX82.2 (setpoint gear stage A to C) DB31, ... DBX16.0 - 16.2 (actual gear stage A to C) DB31, ... DBX16.3 (gear is changed)

DB31, ... DBX83.0	**Speed limit exceeded**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The actual speed exceeds the maximum spindle speed: MD35100 $MA_SPIND_VELO_ LIMIT by more than the spindle speed tolerance: MD35150 $MA_SPIND_DES_VELO_TOL
Corresponding to ...	MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance) MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

DB31, ... DBX83.1	**Setpoint speed limited (programmed speed too high)**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The effective setpoint speed exceeds the actual max. limit value. The setpoint speed is limited to this limit. Limit values: • MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT (maximum speed of gear stage) • MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed) • DB31, ... DBX3.6 (spindle speed limitation to MD35160 $MA_SPIND_EXTERN_VELO_LIMIT) • G26 (upper spindle speed limitation) • LIMS (speed limitation for the master spindle if G96/G961/G97 is active) • VELOLIM: Programmed spindle speed limitation in the open-loop speed controlled mode • Safety Integrated MD36931 $MA_SAFE_VELO_LIMIT (limit value for safely-reduced speed)
Signal state 0 or edge change 1 → 0	The effective setpoint speed of the spindle is outside the maximum limit value.
Application example(s)	In the PLC user program, it can be identified via the interface signal that the spindle setpoint speed has not been reached. Possible responses: • Indicate that the status is permissible and enable path feed: DB21, ... DBX6.0 = 0 (feed disable) • Disable path feed or entire channel: DB21, ... DBX6.0 = 1 (feed disable) DB31, ... DBX83.5 (spindle in setpoint range) is processed
	Safety Integrated In addition to the limit value MD36931 $MA_SAFE_VELO_LIMIT, depending on the active safety speed level SG1 ... SGn, the following machine data should be taken into account: • MD36932 $MA_ SAFE_VELO_OVR_FACTOR • MD36933 $MA_SAFE_DES_VELO_LIMIT Example: All standard limit values are greater than 1500 rpm. • SG1 is active • MD36932 $MA_ SAFE_VELO_OVR_FACTOR[<SG1>] = 1111.11111 [rev/min] • MD36933 $MA_SAFE_DES_VELO_LIMIT[<SG1>] = 90% Programming: M3 S1500 The speed setpoint is limited to 1000 rev/min (MD36932 * MD36933). DB31, ... DBX83.1 = 1
Corresponding to ...	DB21, ... DBX6.0 (feed disable) DB31, ... DBX4.3 (feed / spindle stop) DB31, ... DBX83.5 (spindle in setpoint range)

DB31, ... DBX83.2	**Setpoint speed increased (programmed speed too low)**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The effective setpoint speed is below the current min. limit value. The setpoint speed is limited to this limit. Limit values: • MD35120 $MA_GEAR_STEP_MIN_VELO (minimum speed for automatic gear stage selection M40) • MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT (minimum speed of the gear stage) • G25 (lower spindle speed limitation)
Signal state 0 or edge change 1 → 0	The set speed of the spindle is outside the minimum limit value.
Application example(s)	In the PLC user program, it can be identified via the interface signal that the spindle setpoint speed has not been reached. Possible responses: • Indicate that the status is permissible and enable path feed: DB21, ... DBX6.0 = 0 (feed disable) • Disable path feed or entire channel: DB21, ... DBX6.0 = 1 (feed disable) DB31, ... DBX83.5 (spindle in setpoint range) is processed The interface signal indicates if the programmed set speed is unattainable. The feed can be enabled nonetheless by means of the PLC user program. The PLC user program can flag this state as permissible and enable the path feed, or it can disable the path feed or the complete channel, IS:
Corresponding to ...	DB21, ... DBX6.0 (feed disable) DB31, ... DBX4.3 (feed / spindle stop) DB31, ... DBX83.5 (spindle in setpoint range)

DB31, ... DBX83.5	**Spindle in setpoint range**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The actual speed of the spindle deviates from the set speed by less than the spindle speed tolerance: MD35150 $MA_SPIND_DES_VELO_TOL.
Signal state 0 or edge change 1 → 0	The actual speed of the spindle deviates from the set speed by more than the spindle speed tolerance: MD35150 $MA_SPIND_DES_VELO_TOL. Normal status during the acceleration/deceleration phase of the spindle.
Signal irrelevant for ...	... All spindle modes except for control mode (speed mode).

DB31, ... DBX83.5	Spindle in setpoint range
Application example(s)	Feed enable in the channel only at the end of the acceleration phase of the spindle: IF (DB31, ... DBX83.5 (spindle in setpoint range) == 1) THEN (DB21, ... DBX6.0 (feed disable) = 0) ELSE (DB21, ... DBX6.0 (feed disable) = 1) Note: With the feed disable, the positioning axes are also stopped.
Corresponding to ...	MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START MD35500 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance)

DB31, ... DBX83.7	Actual direction of rotation clockwise
Edge evaluation: Yes	Signal(s) updated: Cyclically
	Interface signal is only valid if the spindle is rotating: DB31, ... DBX61.4 (axis/spindle stationary) == 0 The actual direction of rotation is derived from the position measuring encoder.
Signal state 1 or edge change 0 → 1	Actual direction of rotation: Right
Signal state 0 or edge change 1 → 0	Actual direction of rotation: Left
Signal irrelevant for ...	• Spindle is stationary: DB31, ... DBX61.4 (axis/spindle stationary) == 1 • Spindles without position measuring encoder
Corresponding to ...	DB31, ... DBX61.4 (axis/spindle stationary)

DB31, ... DBX84.3	Rigid tapping active
Edge evaluation: Yes	Signal(s) updated: Cyclically
	The spindle is internally switched to axis mode by the "Rigid tapping" function (G331/G332). This results in a reaction to or updating of the spindle-specific interface signals: • DB31, ... DBX2.2 (spindle reset) • DB31, ... DBX16.4 - DBX16.5 (synchronize spindle) • DB31, ... DBX17.6 (invert M3/M4) • DB31, ... DBX83.5 (spindle in set range) • DB31, ... DBX83.1 (programmable speed too high)
Signal state 1 or edge change 0 → 1	• Rigid tapping active.
Application example(s)	Notice! If the following signals are set during rigid tapping, the thread will be destroyed: • DB11, … DBX0.7 (mode group reset) = 1 • DB21, ... DBX7.7 (channel reset) = 1 • DB31, ... DBX2.1 (controller enable) = 0 • DB31, ... DBX4.3 (feed stop) = 1

DB31, ... DBX84.5	Active spindle mode: Positioning mode
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Positioning mode (SPOS or SPOSA) is active.
Corresponding to ...	DB31, ... DBX84.7 (spindle mode control mode) DB31, ... DBX84.6 (spindle mode oscillation mode)

DB31, ... DBX84.6	Active spindle mode: Oscillation mode
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Oscillation mode is active. Note: The spindle changes automatically to oscillation mode if there is a gear change.
Corresponding to ...	DB31, ... DBX84.7 (spindle mode control mode) DB31, ... DBX84.5 (spindle mode positioning mode) DB31, ... DBX82.3 (change gear)

DB31, ... DBX84.7	**Active spindle mode: Control mode**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The spindle is in control mode with the following functions: • Spindle direction of rotation specification M3/M4 or spindle stop M5 • M41...M45, or automatic gear stage change M40
Corresponding to ...	DB31, ... DBX84.6 (spindle mode oscillation mode) DB31, ... DBX84.5 (spindle mode positioning mode)

DB31, ... DBX85.5	**Spindle in position**
Edge evaluation: Yes	Signal(s) updated: Cyclically
	The interface signal is processed exclusively with the function spindle positioning. This includes: • SPOS, SPOSA and M19 in the part program • SPOS and M19 in synchronized actions • Spindle positioning, using FC18 • Spindle positioning via PLC interface (DB31, ... DBX30.4)
Signal state 1 or edge change 0 → 1	Requirement for the output of the DB31, ... DBX85.5 signal (spindle in position) is reaching the "Exact stop fine". DB31, ... DBX60.7 (exact stop fine) = 1 Additionally, the last programmed spindle position must have been reached on the setpoint side. If the spindle is already at the programmed position after a positioning, then signal DB31,... DBX85.5 (spindle in position) is set.
Signal state 0 or edge change 1 → 0	When signal DB31, ... DBX60.7 is withdrawn (exact stop fine), then signal DB31, ... DBX85.5 (spindle in position) is also always reset.
Application example(s)	Spindle in position for the tool change If the machine operator interrupts the tool change cycle (e.g. with NC stop, NC stop axis plus spindle, mode group stop, etc.), then NC/PLC interface signal DB31, ... DBX85.5 can be used to query that the position has been reached with which the spindle should enter the tool changer.
Corresponding to ...	DB31, ... DBX60.7 (exact stop fine)

DB31, ... DBB86	**M function for spindle**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	From the NCK one of the following M functions: `M3`, `M4`, `M5`, `M19`, `M70` is output to the PLC. The output is performed via: See "Corresponds to ..." below
Corresponding to ...	DB31, ... DBB86 - DBB87 (M function for spindle), axis-specific DB21, ... DBB58, DBB68 - DBB97 (M function for spindle), channel-specific

DB31, ... DBB88	**S function for spindle**
Edge evaluation: Yes	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	An S function was output from the NCK to the PLC. The output occurs by means of: See "Corresponds to ..." below The following S functions are output here: • S.... as spindle speed in rpm (programmed value) • S .... as constant cutting rate in m/min or ft/min The following S functions are **not** output here: • S.... as the programmed spindle speed limiting `G25` • S.... as the programmed spindle speed limiting `G26` • S .... as spindle speed in rpm if a spindle was not defined in the controller • S.... as the dwell time in spindle revolutions
Corresponding to ...	DB31, ... DBB88 - DBB91 (S function for spindle), axis-specific DB21, ... DBB60, DBB98 - DBB115 (S function for spindle), channel-specific

Spindle with SMI 24 (Weiss spindle)

DB31, ... DBX132.0	**Sensors available**
Edge evaluation: No	Signal(s) updated: Power-up
Signal state 1	The sensor required for spindles with SMI 24 is available.
Signal state 0	The sensor required for spindles with SMI 24 is not available.
Corresponding to ...	DB31, … DBX132.1: Sensor S1 (clamped state) is available DB31, … DBX132.4: Sensor S4 is available DB31, … DBX132.5: Sensor S5 is available

DB31, ... DBX132.1	Sensor S1 available (clamped state)
Edge evaluation: No	Signal(s) updated: Power-up
Signal state 1	Sensor S1 is available.
Signal state 0	Sensor S1 is not available.
Corresponding to ...	DB31, … DBW134 (status of the clamping system) DB31, … DBW136 (analog value: clamped state)

DB31, ... DBX132.4	Sensor S4 available (piston end position)
Edge evaluation: No	Signal(s) updated: Power-up
Signal state 1	Sensor S4 is available.
Signal state 0	Sensor S4 is not available.
Corresponding to ...	DB31, … DBX138.4 (Sensor S4: piston end position)

DB31, ... DBX132.5	Sensor S5 available (angular position of the motor shaft)
Edge evaluation: No	Signal(s) updated: Power-up
Signal state 1	Sensor S5 is available.
Signal state 0	Sensor S5 is not available.
Corresponding to ...	DB31, … DBX138.5 (Sensor S5: angular position of the motor shaft)

DB31, ... DBX133.2	State value is generated, speed limitation p5043 is active
Edge evaluation: No	Signal(s) updated: Power-up
Signal state 1	The state value is generated and the speed limitations from drive parameter p5043 are active.
Signal state 0	The state value is not generated and the speed limitations from drive parameter p5043 are not active.
Note	When generating the state value, the analog voltage values of sensor S1 are transformed into discrete state values of drive parameter r5001.
Corresponding to ...	DB31, … DBX134 (clamped state) Drive parameters: r5001 System variable: $VA_MOT_CLAMPING_STATE[<axis>] OPI variables: vaMotClampingState

DB31, ... DBW134	**Status of the clamping system (sensor S1)**
Edge evaluation: No	Signal(s) updated: Cyclically
	Depending on the position of the clamping device, sensor S1 supplies an analog voltage value. To simplify the evaluation of the clamped state, the analog voltage of sensor module SMI 24 is converted into a state value. The state values correspond to certain voltage ranges. The voltage ranges can be set via: Drive parameter p5041[0...5].
	State value / **Clamped state**
	0 / Sensor S1 not available or state values inactive
	1 / State initialization running
	2 / Released with signal (error state)
	3 / Released
	4 / Clamping with tool
	5 / Releasing with tool
	6 / Releasing without tool
	7 / Clamped with tool AND S4 == 0
	8 / Clamped with tool AND S4 == 1
	9 / Clamping without tool
	10 / Clamped without tool
	11 / Clamped with signal (error state)
Corresponding to ...	DB31, … DBW136 (analog value: clamped state) Drive parameters: p5041[0...5], p5043[0...6]

DB31, ... DBW136	**Analog measured value: Of the clamping system**
Edge evaluation: No	Signal(s) updated: Cyclically
	Sensor S1 supplies an analog voltage value: 0 - 10 V. The analog value of the clamped state is mapped to: 0 - 10000 increments, resolution 1 mV Note SIMATIC S7 input module: 0 - 27648 increments, resolution 0.36 mV Adaptation factor if you change to a spindle with SMI 24: 2.7648
Corresponding to ...	DB31, … DBW134 (clamped state) Drive parameters: p5041[0...5], p5043[0...6]

DB31, ... DBX138.4	**Sensor S4, piston end position**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The piston is in position, i.e. the piston is free to move
Signal state 0	The piston is not in position
Corresponding to ...	DB31, … DBX132.4 (Sensor S4 available)

DB31, ... DBX138.5	**Sensor S5, angular position of the motor shaft**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1	The motor shaft is in position (requirement: The spindle is stationary)
Signal state 0	The motor shaft is not aligned
Corresponding to ...	DB31, … DBX132.5 (sensor S5 available)

18.12 Feeds (V1)

18.12.1 Signals to channel (DB21, ...)

DB21, ... DBX0.6	Activate dry run feed	
Edge evaluation: Yes		Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The dry run feedrate defined using the setting data: SD42100 $SC_DRY_RUN_FEED is used instead of the programmed feed (for G01, G02, G03) if the dry run feedrate is greater than that programmed. The dry run feedrate is effective after the reset state. This interface signal is evaluated at NC start when the channel is in the "reset" state. The dry run feed can be activated from the PLC or operator panel. When selected from the operator panel front, the PLC interface signal: DB21, ... DBX24.6 (dry run feed selected) is set and transferred from the PLC basic program to the interface signal: DB21, ... DBX0.6 (activate dry run feed). When selected using the PLC, the IS "activate dry run feed" should be set from the PLC user program.	
Signal state 0 or edge change 1 → 0	The programmed feed is used. Effective after the reset state.	
Application example(s)	Testing a workpiece program with an increased feedrate.	
Special cases, errors, ...	If the signal changes to "0" within a G33 block, the programmed feed is not activated until the end of the block is reached, since an NC stop was not triggered.	
Corresponding to ...	DB21, ... DBX24.6 (dry run feedrate selected) SD42100 $SC_DRY_RUN_FEED (dry run feedrate)	

DB21, ... **DBB4**	**Feedrate override**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The feedrate override can be defined via the PLC in binary or Gray coding. With binary coding, the feed value is interpreted in %. 0% to 200% feed changes are possible, in accordance with the binary value in the byte. The following permanent assignment applies: Code — Feed rate override factor 00000000 — 0.00 ≙ 0% 00000001 — 0.01 ≙ 1% 00000010 — 0.02 ≙ 2% 00000011 — 0.03 ≙ 3% . — . 11001000 — 2.00 ≙ 200% Binary values > 200 are limited to 200%. The machine data: MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch) can be used to additionally limit the maximum feedrate override.

DB21, ... DBB4	Feedrate override
	In gray coding, the following codes are assigned to the individual switch settings: Table: Gray coding for feed rate override Switch position / Code / Feed rate override factor (standard values) 1 / 00001 / 0.0 2 / 00011 / 0.01 3 / 00010 / 0.02 4 / 00110 / 0.04 5 / 00111 / 0.06 6 / 00101 / 0.08 7 / 00100 / 0.10 8 / 01100 / 0.20 9 / 01101 / 0.30 10 / 01111 / 0.40 11 / 01110 / 0.50 12 / 01010 / 0.60 13 / 01011 / 0.70 14 / 01001 / 0.75 15 / 01000 / 0.80 16 / 11000 / 0.85 17 / 11001 / 0.90 18 / 11011 / 0.95 19 / 11010 / 1.00 20 / 11110 / 1.05 21 / 11111 / 1.10 22 / 11101 / 1.15 23 / 11100 / 1.20 24 / 10100 / 1.20 25 / 10101 / 1.20 26 / 10111 / 1.20 27 / 10110 / 1.20 28 / 10010 / 1.20 29 / 10011 / 1.20 30 / 10001 / 1.20 31 / 10000 / 1.20 The factors listed in the table for the feedrate override are stored in the machine data: MD12030 $MN_OVR_FACTOR_FEEDRATE [n]. The table contains the default settings. The number of possible switch settings for standard machine panels is described in the Configuration Manual for SINUMERIK 840D.
Corresponding to ...	DB21, ... DBX6.7 (feedrate override active) MD12030 $MN_OVR_FACTOR_FEEDRATE [n] (evaluation of the path feedrate override switch) MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

DB21, ... **DBB5**	**Rapid traverse override**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The rapid traverse override can be entered via the PLC in either the binary or Gray code. For binary coding, the rapid traverse override is interpreted as a %. 0% to 100% feed changes are possible, in accordance with the binary value in the byte. The following permanent assignment applies: Code — Rapid traverse override factor 00000000 — 0.00 ≙ 0% 00000001 — 0.01 ≙ 1% 00000010 — 0.02 ≙ 2% 00000011 — 0.03 ≙ 3% . — . 11001000 — 1.00 ≙ 100% Binary values > 100 are limited to 100%. Using the machine data: MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch), the maximum rapid traverse override can be additionally limited.

DB21, ... DBB5	Rapid traverse override
	In gray coding, the following codes are assigned to the individual switch settings:

Table: Gray coding for rapid traverse override

Switch position	Code	Rapid traverse override factor (standard values)
1	00001	0.0
2	00011	0.01
3	00010	0.02
4	00110	0.04
5	00111	0.06
6	00101	0.08
7	00100	0.10
8	01100	0.20
9	01101	0.30
10	01111	0.40
11	01110	0.50
12	01010	0.60
13	01011	0.70
14	01001	0.75
15	01000	0.80
16	11000	0.85
17	11001	0.90
18	11011	0.95
19	11010	1.00
20	11110	1.00
21	11111	1.00
22	11101	1.00
23	11100	1.00
24	10100	1.00
25	10101	1.00
26	10111	1.00
27	10110	1.00
28	10010	1.00
29	10011	1.00
30	10001	1.00
31	10000	1.00

The factors listed in the table for the rapid traverse override are stored in the machine data:
MD12050 $MN_OVR_FACTOR_RAPID_TRA[n].

The table contains the default settings.

The number of possible switch settings for standard machine panels is described in the Configuration Manual for SINUMERIK 840D.

Corresponding to ...	DB21, ... DBX6.6 (rapid traverse override active) MD12050 $MN_OVR_FACTOR_RAPID_TRA[n] (evaluation of the path feedrate override switch) MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

DB21, ... DBX6.0	**Feed disable**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The signal is active in one channel in all operating modes. The signal disables the feed for all of the axes (geometry and synchronized) that interpolate relative to one another as long as G33 (thread) is not active. All axes are brought to a standstill but still maintaining the path contour. When the feed disable is canceled (0 signal), the interrupted part program is continued. The signal triggers a feed disable for all positioning axes. This signal brings all traversing axes to a standstill with controlled braking (ramp stop). No alarm is output. The position control is retained, i.e. the following error is eliminated. If a travel request is issued for an axis with an active "Feed disable", then this is kept. The queued travel request is executed immediately when the "Feed disable" is canceled. If the axis is interpolating in relation to others, this also applies to these axes.
Signal state 0 or edge change 1 → 0	The feedrate is enabled for all axes of the channel. If a travel request ("travel command") exists for an axis or group of axes when the "feed disable" is canceled, then this is executed immediately.
Application example(s)	Stopping machining by selecting FEED OFF on the machine control panel.
Special cases, errors, ...	The feed disable is inactive when G33 is active.

DB21, ... DBX6.6	**Rapid traverse override active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The rapid traverse override between 0 and a maximum of 100% entered in the PLC interface is channel-specific. The override factor is defined using the machine data: MD12040 $MN_OVR_RAPID_IS_GRAY_CODE (rapid traverse override switch gray coded) and MD12050 $MN_OVR_FACTOR_RAPID_TRA [n] (evaluation of the rapid traverse override switch).
Signal state 0 or edge change 1 → 0	The rapid traverse override entered at the PLC interface is ignored. When the rapid traverse override is inactive, the NC always uses 100% as the internal override factor. Exceptions are the zero setting for a binary interface and the 1st switch setting for a Gray-coded interface. In these cases, the override factors entered at the PLC interface are used. With a binary interface, the override factor = 0. With a gray-coded interface, the value entered in the machine data for the 1st switch setting is output as the override value.
Application example(s)	The override value is generally selected using the rapid traverse override switch on the machine control panel. Using the interface signal: DB21, ... DBX6.6 (rapid traverse override active), the rapid traverse override switch can be enabled from the PLC user program while commissioning a new NC program, e.g. using the key-operated switch.

DB21, ... DBX6.6	**Rapid traverse override active**
Special cases, errors, ...	The rapid traverse override is inactive when G33, G63, G331, G332 are active.
Corresponding to ...	DB21, ... DBB5 (rapid traverse override)

DB21, ... DBX6.7	**Feedrate override active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The feedrate override between 0 and a maximum of 200% entered at the PLC interface is active for the path feedrate and therefore automatically for the related axes. In JOG mode, the feedrate override acts directly on the axes. The override factor is entered using the machine data: MD12020 $MN_OVR_FEED_IS_GRAY_CODE (path feedrate override factor, gray-coded) and MD12030 $MN_OVR_FACTOR_FEEDRATE [n] (evaluation of the path feedrate override switch)
Signal state 0 or edge change 1 → 0	The feedrate override entered at the PLC interface is ignored. When the feedrate override is inactive, the NC always uses 100% as the internal override factor. Exceptions are the zero setting for a binary interface and the 1st switch setting for a Gray-coded interface. In these cases, the override factors entered in the PLC interface are used. With a binary interface, the override factor = 0. With a gray-coded interface, the value entered in the machine data for the 1st switch setting is output as the override value.
Application example(s)	The override value is generally selected using the feedrate override switch on the machine control panel. Using the interface signal: DB21, ... DBX6.7 (feedrate override active), the feedrate override switch can be enabled from the PLC user program while commissioning a new NC program, e.g. using the key-operated switch.
Special cases, errors, ...	The feedrate override is inactive when G33, G63, G331, G332 are active.
Corresponding to ...	DB21, ... DBB4 (feedrate override)

DB21, ... DBX12.3, DBX16.3, DBX20.3	Feed stop (Geometry axis 1 to 3)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The signal is only active in JOG mode. The signal stops the feed of the geometry axis. This signal brings all traversing axes to a standstill with controlled braking (ramp stop). No alarm is output. The position control is retained, i.e. the following error is eliminated. If, for a geometry axis, a travel request is issued with an active "feed stop", the request is kept. This queued travel request is executed immediately after the "feed stop" is canceled.
Signal state 0 or edge change 1 → 0	The feed is enabled for the geometry axis. If, for the geometry axis, a travel request ("travel command") is active when the "feed stop" is canceled, this is executed immediately.

DB21, ... DBX24.6	Dry run feedrate selected
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Dry run feedrate is selected. Instead of the programmed feedrate, the dry run feedrate entered in setting data: SD42100 $SC_DRY_RUN_FEED is active. When activated from the operator panel, the dry run feed signal is automatically entered in the PLC interface and transmitted by the PLC basic program to the PLC interface signal: DB21, ... DBX0.6 (active dry run feed).
Signal state 0 or edge change 1 → 0	Dry run feedrate is not selected. The programmed feedrate is active.
Corresponding to ...	DB21, ... DBX0.6 (activate dry run feed) SD42100 $SC_DRY_RUN_FEED (dry run feedrate)

DB21, ... DBX25.3	**Feedrate override selected for rapid traverse**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The feedrate override switch should also be active as rapid traverse override switch. Override values above 100% are limited to the maximum value for 100% rapid traverse override. The interface signal: DB21, ... DBX25.3 (feedrate override for rapid traverse selected) is automatically entered into the PLC interface from the operator panel and transferred from the PLC basic program to the PLC interface signal: DB21, ... DBX6.6 (rapid traverse override active). Further, the interface signal: DB21, ... DBB4 (feedrate override) is copied from the PLC basic program to the interface signal: DB21, ... DBB5 (rapid traverse override).
Signal state 0 or edge change 1 → 0	The feedrate override switch should not be activated as rapid traverse override switch.
Application example(s)	The signal is used when no separate rapid traverse override switch is available.

DB 21, ... DBX29.0 - DBX29.3	**Activate fixed feedrate 1 - 4 for path/geometry axes**
Edge evaluation: No	Signal(s) updated: Cyclically
Description	These signals are used to select/de-select the function "fixed feed" and define which fixed feed should be effective for path/geometry axes.
Corresponding to ...	MD12202 $MN_PERMANENT_FEED[ŋ] MD12200 $MN_RUN_OVERRIDE_0

Bit 3	Bit 2	Bit 1	Bit 0	Meaning
0	0	0	0	Fixed feed is de-selected
0	0	0	1	Fixed feed 1 is selected
0	0	1	0	Fixed feed 2 is selected
0	1	0	0	Fixed feed 3 is selected
1	0	0	0	Fixed feed 4 is selected

18.12.2 Signals to axis/spindle (DB31, ...)

DB31, ... DBB0	**Feedrate override (axis-specific)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The axis-specific feedrate override can be defined via the PLC in binary or Gray coding. With binary coding, the feed value is interpreted in %. 0% to 200% feed changes are possible, in accordance with the binary value in the byte. The following permanent assignment applies: Code: 00000000 → Axis-specific feed rate override factor: 0.00 ≙ 0% 00000001 → 0.01 ≙ 1% 00000010 → 0.02 ≙ 2% 00000011 → 0.03 ≙ 3% 01100100 → 1.00 ≙ 100% 11001000 → 2.00 ≙ 200%

DB31, ... DBB0	**Feedrate override (axis-specific)**
	Binary values > 200 are limited to 200%. Using the machine data: MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch) the maximum axis-specific feedrate override can be additionally limited. In gray coding, the following codes are assigned to the individual switch settings:

Table: Gray coding for axis-specific feed rate override

Switch position	Code	Axial feed rate override factor (standard values)
1	00001	0.0
2	00011	0.01
3	00010	0.02
4	00110	0.04
5	00111	0.06
6	00101	0.08
7	00100	0.10
8	01100	0.20
9	01101	0.30
10	01111	0.40
11	01110	0.50
12	01010	0.60
13	01011	0.70
14	01001	0.75
15	01000	0.80
16	11000	0.85
17	11001	0.90
18	11011	0.95
19	11010	1.00
20	11110	1.05
21	11111	1.10
22	11101	1.15
23	11100	1.20
24	10100	1.20
25	10101	1.20
26	10111	1.20
27	10110	1.20
28	10010	1.20
29	10011	1.20
30	10001	1.20
31	10000	1.20

	The factors listed in the table for the axial feedrate override are stored in the NC-specific machine data: MD12010 $MN_OVR_FACTOR_AX_SPEED [n] The table contains the default settings. The number of possible switch settings for standard machine panels is described in the Configuration Manual for SINUMERIK 840D.
Corresponding to ...	DB31, ... DBX1.7 (override effective) MD12010 $MN_OVR_FACTOR_AX_SPEED [n] (evaluation of the axis feedrate override switch) MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

DB31, ... DBX1.7	**Override active**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	**Feedrate override active:** The axis-specific feedrate override between 0 and a maximum of 200% entered in the PLC interface is used. The override factor is defined using the machine data: MD12000 $MN_OVR_AX_IS_GRAY_CODE (axis feedrate override switch gray coded) and MD12010 $MN_OVR_FACTOR_AX_SPEED [n] (evaluation of the axis feedrate override switch). **Spindle override active:** The spindle override - input at the PLC interface - of 0 to a maximum of 200% is taken into account. The override factor is entered using the machine data: MD12060 $MN_OVR_SPIND_IS_GRAY_CODE (spindle override switch, Gray coded) and MD12070 $MN_OVR_FACTOR_SPIND_SPEED [n] (evaluation of the spindle override switch).
Signal state 0 or edge change 1 → 0	The existing axis-specific feedrate override or spindle override is not active. If the feedrate override is inactive, "100%" is used as the internal override factor. Exceptions are the zero setting for a binary interface and the 1st switch setting for a Gray-coded interface. In these cases, the override factors entered at the PLC interface are used. With a binary interface, the override factor = 0. With a gray-coded interface, the value entered in the machine data for the 1st switch setting is output as the override value.
Application example(s)	The override value is generally specified using the axis-specific feedrate override switch or the spindle override switch on the machine control panel. The "feedrate override active" signal can be used to enable the feedrate override switch from the PLC user program, e.g. using the key-operated switch when commissioning a new NC program.
Special cases, errors, ...	The spindle override is always accepted with 100% in the spindle "Oscillation mode". The spindle override acts on the programmed values before the limits (e.g. `G26`, `LIMS`...) intervene. The feedrate override is ineffective for: • Active `G33` • Active `G63` (the override is defined in the NC at 100%) • Active `G331`, `G332` (the override is defined in the NC at 100%) The spindle override is inactive for: • Active `G63` (the override is defined in the NC at 100%)
Corresponding to ...	DB31, ... DBB0 (feed/spindle override)

DB31, ... DBX3.2 - DBX3.5	Activate fixed feedrate 1 - 4 for machine axes
Edge evaluation: No	Signal(s) updated: Cyclically
Description	These signals are used to select/de-select the function "fixed feed" and define which fixed feed should be effective for machine axes.

Bit 5	Bit 4	Bit 3	Bit 2	Meaning
0	0	0	0	Fixed feed is de-selected
0	0	0	1	Fixed feed 1 is selected
0	0	1	0	Fixed feed 2 is selected
0	1	0	0	Fixed feed 3 is selected
1	0	0	0	Fixed feed 4 is selected

Corresponding to ...	MD12202 $MN_PERMANENT_FEED[n] MD12200 $MN_RUN_OVERRIDE_0

DB31, ... DBX4.3	Feed stop / spindle stop (axis-specific)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The signal is active in all modes. **Feed stop:** The signal triggers a "feed stop" for the axis. This signal brings all traversing axes to a standstill with controlled braking (ramp stop). No alarm is output. The signal triggers a "feed stop" for all path axes interpolating relative to each other when the "feed stop" is activated for any one of these path axes. In this case, all the axes are brought to a stop with adherence to the path contour. When the "feed stop" signal is canceled, execution of the interrupted part program is resumed. The position control is retained, i.e. the following error is eliminated. If a travel request is issued for an axis with an active "feed stop", this is kept. This queued travel request is executed immediately after the "feed stop" is canceled. If the axis is interpolating in relation to others, this also applies to these axes. **Spindle stop:** The spindle is brought to a standstill along the acceleration characteristic. In the positioning mode, when the "Spindle stop" signal is set positioning is interrupted. The above response applies with respect to individual axes.
Signal state 0 or edge change 1 → 0	**Feed stop:** The feedrate is enabled for the axis. If a travel request ("travel command") is active when the "feed stop" is canceled, this is executed immediately. **Spindle stop:** The speed is enabled for the spindle. The spindle is accelerated to the previous speed setpoint with the acceleration characteristic or, in positioning mode, positioning is resumed.

DB31, ... DBX4.3	Feed stop / spindle stop (axis-specific)
Application example(s)	**Feed stop:** The traversing motion of the machine axes is not started with "feed stop", if, for example, certain operating states exist at the machine that do not permit the axes to be moved (e.g. a door is not closed). **Spindle stop:** In order to change a tool To enter help functions (M, S, H, T, D and F functions) during setup.
Special cases, errors, ...	Spindle stop is inactive when `G331`, `G332` are active.

DB31, ... DBB19	Spindle override	
Edge evaluation: No		Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The spindle override can be defined via the PLC in binary or Gray coding. The override value determines the percentage of the programmed speed setpoint that is issued to the spindle. With binary coding, the override is interpreted in %. 0% to 200% feed changes are possible, in accordance with the binary value in the byte. The following permanent assignment applies: Code — Spindle override factor 00000000 — 0.00 ≙ 0% 00000001 — 0.01 ≙ 1% 00000010 — 0.02 ≙ 2% 00000011 — 0.03 ≙ 3% . . . 01100100 — 1.00 ≙ 100% . . . 11001000 — 2.00 ≙ 200% Binary values > 200 are limited to 200%. The machine data: MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch) can be used to additionally limit the maximum spindle override.	

DB31, ... DBB19	Spindle override
Signal state 1 or edge change 0 → 1	In gray coding, the following codes are assigned to the individual switch settings: Table: Gray coding for spindle override Switch position / Code / Spindle override factor (standard values) 1 / 00001 / 0.5 2 / 00011 / 0.55 3 / 00010 / 0.60 4 / 00110 / 0.65 5 / 00111 / 0.70 6 / 00101 / 0.75 7 / 00100 / 0.80 8 / 01100 / 0.85 9 / 01101 / 0.90 10 / 01111 / 0.95 11 / 01110 / 1.00 12 / 01010 / 1.05 13 / 01011 / 1.10 14 / 01001 / 1.15 15 / 01000 / 1.20 16 / 11000 / 1.20 17 / 11001 / 1.20 18 / 11011 / 1.20 19 / 11010 / 1.20 20 / 11110 / 1.20 21 / 11111 / 1.20 22 / 11101 / 1.20 23 / 11100 / 1.20 24 / 10100 / 1.20 25 / 10101 / 1.20 26 / 10111 / 1.20 27 / 10110 / 1.20 28 / 10010 / 1.20 29 / 10011 / 1.20 30 / 10001 / 1.20 31 / 10000 / 1.20 The factors listed in the table for spindle override are stored in the machine data: MD12070 $MN_OVR_FACTOR_SPIND_SPEED [n] The table contains the default settings. The number of possible switch settings for standard machine panels is described in the Configuration Manual for SINUMERIK 840D.
Corresponding to ...	DB31, ... DBX1.7 (override active) MD12070 $MN_OVR_FACTOR_SPIND_SPEED [n] (evaluation of the spindle override switch) MD12100 $MN_FACTOR_LIMIT_BIN (limit for binary-coded override switch)

18.12.3 Signals from axis/spindle (DB31, ...)

DB31, ... DBX62.2	Revolutional feed rate active
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	For programming of G95 (revolutional feed rate) in JOG mode or automatic mode.
Corresponding to	SD41100 $SN_JOG_REV_IS_ACTIVE (revolutional feed rate for JOG active) SD42600 $SC_JOG_FEED_PER_REV_SOURCE (in the JOG mode revolutional feed rate for geometry axes, on which the frame with rotation acts) SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (revolutional feed rate for position axes/spindles) MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feed rate for JOG with rapid traverse override) MD32050 $MA_JOG_REV_VELO (revolutional feed rate for JOG mode)

DB31, ... DBB78 - DBB81	F function for positioning axis
Edge evaluation: no	Signal(s) updated: cyclic
Signal state 1 or edge change 0 → 1	The F value of a positioning axis programmed in the current block is entered in the axisspecific PLC interface signal. The assignment between the DB number and machine axis number is established using the axis name. The value is retained until it is overwritten by another. Format: Binary number in real format.
Application example(s)	Modification of the programmed F value by the PLC, e.g. by overwriting the selected axisspecific feed rate override.
Corresponding to	MD22240 $MC_AUXFU_F_SYNC_TYPE (output time F functions)

Appendix

A

A.1 List of abbreviations

A	
O	Output
ADI4	(Analog drive interface for 4 axes)
AC	Adaptive Control
ALM	Active Line Module
ARM	Rotating induction motor
AS	PLC
ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit: User switching circuit
ASUB	Asynchronous subprogram
AUXFU	Auxiliary Function
STL	Statement List
UP	User Program

B	
BA	Mode
BAG	Mode group
BCD	Binary Coded Decimals: Decimal numbers encoded in binary code
BERO	Proximity limit switch with feedback oscillator
BI	Binector Input
BICO	Binector Connector
BIN	Binary files
BIOS	Basic Input Output System
BCS	Basic Coordinate System
BO	Binector Output
OPI	Operator Panel Interface

C	
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CC	Compile Cycle
CI	Connector Input
CF Card	Compact Flash Card

CNC	Computerized Numerical Control
CO	Connector Output
CoL	Certificate of License
COM	Communication
CPA	Compiler Projecting Data: Configuring data of the compiler
CRT	Cathode Ray Tube
CSB	Central Service Board: PLC module
CU	Control Unit
CP	Communication Processor
CPU	Central Processing Unit
CR	Carriage Return
CTS	Clear To Send: Ready to send signal for serial data interfaces
CUTCOM	Cutter radius Compensation: Tool radius compensation

D	
DAC	Digital-to-Analog Converter
DB	Data Block (PLC)
DBB	Data Block Byte (PLC)
DBD	Data Block Double word (PLC)
DBW	Data Block Word (PLC)
DBX	Data block bit (PLC)
DDE	Dynamic Data Exchange
DIN	Deutsche Industrie Norm
DIO	Data Input/Output: Data transfer display
DIR	Directory
DLL	Dynamic Link Library
DO	Drive Object
DPM	Dual Port Memory
DPR	Dual Port RAM
DRAM	Dynamic memory (non-buffered)
DRF	Differential Resolver Function (handwheel)
DRIVE-CLiQ	Drive Component Link with IQ
DRY	Dry Run: Dry run feedrate
DSB	Decoding Single Block
DSC	Dynamic Servo Control / Dynamic Stiffness Control
DW	Data Word
DWORD	Double Word (currently 32 bits)

E	
I	Input
I/O	Input/Output
ENC	Encoder: Actual value encoder
EFP	Compact I/O module (PLC I/O module)
ESD	Electrostatic Sensitive Devices
EMC	ElectroMagnetic Compatibility
EN	European standard
EnDat	Encoder interface
EPROM	Erasable Programmable Read Only Memory
ePS Network Services	Services for Internet-based remote machine maintenance
EQN	Designation for an absolute encoder with 2048 sine signals per revolution
ES	Engineering System
ESR	Extended Stop and Retract
ETC	ETC key ">"; softkey bar extension in the same menu

F	
FB	Function Block (PLC)
FC	Function Call (PLC)
FEPROM	Flash EPROM: Read and write memory
FIFO	First In First Out: Memory that works without address specification and whose data is read in the same order in which they was stored
FIPO	Fine interpolator
FPU	Floating Point Unit
CRC	Cutter Radius Compensation
FST	Feed Stop: Feedrate stop
FBD	Function Block Diagram (PLC programming method)
FW	Firmware

G	
GC	Global Control (PROFIBUS: Broadcast telegram)
GEO	Geometry, e.g. geometry axis
GIA	Gear Interpolation Data
GND	Signal Ground
GP	Basic program (PLC)

GS	Gear Stage
GSD	Device master file for describing a PROFIBUS slave
GSDML	Generic Station Description Markup Language: XML-based description language for creating a GSD file
GUD	Global User Data

H	
HEX	Abbreviation for hexadecimal number
AuxF	Auxiliary Function
HLA	Hydraulic linear drive
HMI	Human Machine Interface: SINUMERIK user interface
MSD	Main Spindle Drive
HW	Hardware

I	
IBN	Commissioning
ICA	Interpolatory compensation
IM	Interface Module
IMR	Interface Module Receive: Interface module for receiving data
IMS	Interface Module Send: Interface module for sending data
INC	Increment
INI	Initializing Data
IPO	Interpolator
ISA	Industry Standard Architecture
ISO	International Standardization Organization

J	
JOG	Jogging: Setup mode

K	
K_V	Gain factor of control loop
K_p	Proportional gain
$K_Ü$	Transformation ratio
LAD	Ladder Diagram (PLC programming method)

L	
LAI	Logic Machine Axis Image
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LF	Line Feed
PMS	Position Measuring System
LR	Position controller
LSB	Least Significant Bit
LUD	Local User Data

M	
MAC	Media Access Control
MAIN	Main program (OB1, PLC)
MB	Megabyte
MCI	Motion Control Interface
MCIS	Motion Control Information System
MCP	Machine Control Panel
MD	Machine Data
MDA	Manual Data Automatic: Manual input
MSGW	Message Word
MCS	Machine Coordinate System
MLFB	Machine-readable product code
MM	Motor Module
MPF	Main Program File (NC)
MCP	Machine Control Panel

N	
NC	Numerical Control
NCK	Numerical Control Kernel: NC kernel with block preparation, traversing range, etc.
NCU	Numerical Control Unit: NCK hardware unit
NRK	Name for the operating system of the NCK
IS	Interface Signal
NURBS	Non-Uniform Rational B-Spline
ZO	Zero Offset
NX	Numerical Extension: Axis expansion board

O	
OB	Organization block in the PLC
OEM	Original Equipment Manufacturer
OP	Operator Panel
OPI	Operator Panel Interface
OPT	Options
OLP	Optical Link Plug: Fiber optic bus connector
OSI	Open Systems Interconnection: Standard for computer communications

P	
PIQ	Process Image Output
PII	Process Image Input
PC	Personal Computer
PCIN	Name of the SW for data exchange with the controller
PCMCIA	Personal Computer Memory Card International Association: Plug-in memory card standardization
PCU	PC Unit
PG	Programming device
PKE	Parameter identification: Part of a PIV
PIV	Parameter Identifier Value (parameterization part of a PPO)
PLC	Programmable Logic Control: Adaptation control
PN	PROFINET
PNO	PROFIBUS user organization
PO	POWER ON
POU	Program Organization Unit
POS	Position/positioning
POSMO A	Positioning Motor Actuator
POSMO CA	Positioning Motor Compact AC: Complete drive unit with integrated power and control module as well as positioning unit and program memory; AC infeed
POSMO CD	Positioning Motor Compact DC: Like CA but with DC infeed
POSMO SI	Positioning Motor Servo Integrated: Positioning motor, DC infeed
PPO	Parameter Process data Object: Cyclic data telegram for PROFIBUS DP transmission and "Variable speed drives" profile
PPU	Panel Processing Unit (central hardware for a panel-based CNC, e.g SINUMERIK 828D)
PROFIBUS	Process Field Bus: Serial data bus
PRT	Program Test
PSW	Program control word
PTP	Point-To-Point
PUD	Program global User Data
PZD	Process data: Process data part of a PPO

Q	
QEC	Quadrant Error Compensation

R	
RAM	Random Access Memory: Read/write memory
REF	REFerence point approach function
REPOS	REPOSition function
RISC	Reduced Instruction Set Computer: Type of processor with small instruction set and ability to process instructions at high speed
ROV	Rapid Override: Input correction
RP	R Parameter, arithmetic parameter, predefined user variable
RPA	R Parameter Active: Memory area on the NCK for R parameter numbers
RPY	Roll Pitch Yaw: Rotation type of a coordinate system
RTLI	Rapid Traverse Linear Interpolation:
RTS	Request To Send: Control signal of serial data interfaces
RTCP	Real Time Control Protocol

S	
SA	Synchronized Action
SBC	Safe Brake Control
SBL	Single Block
SBR	Subprogram (PLC)
SD	Setting Data
SDB	System Data Block
SEA	Setting Data Active: Identifier (file type) for setting data
SERUPRO	SEarch RUn by PROgram test
SFB	System Function Block
SFC	System Function Call
SGE	Safety-related input
SGA	Safety-related output
SH	Safe standstill
SIM	Single Inline Module
SK	Softkey
SKP	Skip: Function for skipping a part program block
SLM	Synchronous Linear Motor
SM	Stepper Motor
SMC	Sensor Module Cabinet Mounted
SME	Sensor Module Externally Mounted

SMI	Sensor Module Integrated
SPF	Subprogram File (NC)
PLC	Programmable Logic Controller
SRAM	Static RAM (non-volatile)
TNRC	Tool Nose Radius Compensation
SRM	Synchronous Rotary Motor
LEC	Leadscrew Error Compensation
SSI	Synchronous Serial Interface
SSL	Block search
STW	Control word
GWPS	Grinding Wheel Peripheral Speed
SW	Software
SYF	System Files
SYNACT	SYNchronized ACTion: Synchronized Action

T	
TB	Terminal Board (SINAMICS)
TCP	Tool Center Point
TCP/IP	Transport Control Protocol / Internet Protocol
TCU	Thin Client Unit
TEA	Testing Data Active: Identifier for machine data
TIA	Totally Integrated Automation
TM	Terminal Module (SINAMICS)
TO	Tool Offset
TOA	Tool Offset Active: Identifier (file type) for tool offsets
TRANSMIT	Transform Milling Into Turning: Coordination transformation for milling operations on a lathe
TTL	Transistor-Transistor Logic (interface type)
TZ	Technology cycle

U	
UFR	User Frame: Zero Offset
SR	Subprogram
USB	Universal Serial Bus
UPS	Uninterruptible Power Supply

V	
VDI	Internal communication interface between NCK and PLC
VDI	Verein Deutscher Ingenieure [Association of German Engineers]

VDE	Verband Deutscher Elektrotechniker [Association of German Electrical Engineers]
VI	Voltage Input
VO	Voltage Output
FDD	Feed Drive

W	
SAR	Smooth Approach and Retraction
WCS	Workpiece Coordinate System
T	Tool
TLC	Tool Length Compensation
WOP	Workshop-Oriented Programming
WPD	Workpiece Directory
TRC	Tool Radius Compensation
T	Tool
TO	Tool Offset
TM	Tool Management
TC	Tool change

X	
XML	Extensible Markup Language

Z	
ZOA	Zero Offset Active: Identifier for zero offsets
ZSW	Status word (of drive)

A.2 Documentation overview

General documentation

- SINUMERIK — Sales Brochure
  – SINUMERIK 840D sl
  – SINUMERIK 828D
  – SINUMERIK 828D BASIC T
- SINUMERIK 840D sl 840Di sl 802D sl — Catalog NC 62
- SINUMERIK 828D — Catalog NC 82
- SINAMICS S120 — Catalog PM 21 SIMOTION, SINAMICS S120 and Motors for Production Machines
- SINUMERIK — Configuration Manual EMC Installation Guidelines
- SINUMERIK 840D sl 828D — System Manual Ctrl Energy

User documentation

- SINUMERIK 840D sl — Operating Manual
  – Universal
  – Turning
  – Milling
  – HMI-Advanced
- SINUMERIK 840D sl 828D — Operating Manual
  – Turning
  – Milling
- SINUMERIK 840D sl 828D — Programming Manual
  – Fundamentals
  – Job Planning
  – Measuring Cycles
- SINUMERIK 840D sl 828D 802D sl — Programming Manual
  – ISO Turning
  – ISO Milling
- SINUMERIK 840D sl S120 — Diagnostics Manual
- SINUMERIK 828D S120 — Diagnostics Manual

Manufacturer/service documentation

- SINUMERIK 840D sl — Manual
  – NCU
  – Operator Components and Networking
- SINUMERIK 828D — Manual
  Commissioning Manual
  Service Manual
- SINUMERIK 840D sl — Commissioning Manual
  – CNC: NCK, PLC, Drive
  – Basic Software and Operating Software
  – Basic Software and HMI-Advanced
- SINUMERIK 840D sl S120 — List Manual
  – Part 1
  – Part 2
  - Detailed Machine Data Description
  – System Variables
- SINUMERIK 828D S120 — List Manual
  – Machine Data and Interface Signals
  – Parameter Description
- SINUMERIK 840D sl S120 — System Manual Guidelines for Machine Configuration

Manufacturer/service documentation

- SINUMERIK 840D sl 828D — Function Manual
  – Basic Functions
  – Extended Functions
  – Special Functions
  - Synchronized Actions
  - Tool Management
- SINUMERIK 840D sl 802D sl 828D — Function Manual
  ISO Dialects
- SINAMICS S120 — Function Manual
  Drive Functions
- SINUMERIK 840D sl S120 — Function Manual
  Safety Integrated
- SINUMERIK 828D S120 — Function Manual
  Safety Integrated

Info/Training

- SINUMERIK — Getting Started
  Milling and Turning
- SINUMERIK — Training Documents
  – Easy Milling with ShopMill
  – Easy Turning with ShopTurn
- SINUMERIK — Manuals
  Tool and Mold Making

Electronic documentation

DOConCD

My Documentation Manager

Industry Mall

Glossary

Absolute dimensions

A destination for an axis motion is defined by a dimension that refers to the origin of the currently active coordinate system. See → Incremental dimension

Acceleration with jerk limitation

In order to optimize the acceleration response of the machine whilst simultaneously protecting the mechanical components, it is possible to switch over in the machining program between abrupt acceleration and continuous (jerk-free) acceleration.

Address

An address is the identifier for a certain operand or operand range, e.g. input, output, etc.

Alarms

All → messages and alarms are displayed on the operator panel in plain text with date and time and the corresponding symbol for the cancel criterion. Alarms and messages are displayed separately.

1. Alarms and messages in the part program:

   Alarms and messages can be displayed in plain text directly from the part program.

2. Alarms and messages from the PLC:

   Alarms and messages for the machine can be displayed in plain text from the PLC program. No additional function block packages are required for this purpose.

Archiving

Reading out of files and/or directories on an **external** memory device.

Asynchronous subprogram

Part program that can be started asynchronously to (independently of) the current program status using an interrupt signal (e.g. "Rapid NC input" signal).

Automatic

Operating mode of the controller (block sequence operation according to DIN): Operating mode for NC systems in which a → subprogram is selected and executed continuously.

Auxiliary functions

Auxiliary functions enable → part programs to transfer → parameters to the → PLC, which then trigger reactions defined by the machine manufacturer.

Axes

In accordance with their functional scope, the CNC axes are subdivided into:

- Axes: Interpolating path axes
- Auxiliary axes: Non-interpolating feed and positioning axes with an axis-specific feedrate. Auxiliary axes are not involved in actual machining, e.g. tool feeder, tool magazine.

Axis address

See → Axis identifier

Axis identifier

Axes are identified using X, Y, and Z as defined in DIN 66217 for a dextrorotory, right-angled → coordinate system.

Rotary axes rotating around X, Y, and Z are identified using A, B, and C. Additional axes situated parallel to the specified axes can be designated using other letters.

Axis name

See → Axis identifier

Backlash compensation

Compensation for a mechanical machine backlash, e.g. backlash on reversal for ball screws. Backlash compensation can be entered separately for each axis.

Backup battery

The backup battery ensures that the → user program in the → CPU is stored so that it is safe from power failure and so that specified data areas and bit memory, timers and counters are stored retentively.

Basic axis

Axis whose setpoint or actual value position forms the basis of the calculation of a compensation value.

Basic Coordinate System

Cartesian coordinate system which is mapped by transformation onto the machine coordinate system.

The programmer uses axis names of the basic coordinate system in the → part program. The basic coordinate system exists parallel to the → machine coordinate system if no → transformation is active. The difference between the two coordinate systems lies in the → axis identifiers.

Baud rate

Rate of data transfer (bits/s).

Blank

Workpiece as it is before it is machined.

Block

"Block" is the term given to any files required for creating and processing programs.

Block search

For debugging purposes or following a program abort, the "Block search" function can be used to select any location in the part program at which the program is to be started or resumed.

Booting

Loading the system program after power ON.

C axis

Axis around which the tool spindle describes a controlled rotational and positioning motion.

C spline

The C spline is the most well-known and widely used spline. The transitions at the interpolation points are continuous, both tangentially and in terms of curvature. 3rd order polynomials are used.

Channel

A channel is characterized by the fact that it can process a → part program independently of other channels. A channel exclusively controls the axes and spindles assigned to it. Part program runs of different channels can be coordinated through → synchronization.

Circular interpolation

The → tool moves on a circle between specified points on the contour at a given feedrate, and the workpiece is thereby machined.

CNC

See → NC

COM

Component of the NC for the implementation and coordination of communication.

Compensation axis

Axis with a setpoint or actual value modified by the compensation value

Compensation table

Table containing interpolation points. It provides the compensation values of the compensation axis for selected positions on the basic axis.

Compensation value

Difference between the axis position measured by the encoder and the desired, programmed axis position.

Continuous-path mode

The objective of continuous-path mode is to avoid substantial deceleration of the → path axes at the part program block boundaries and to change to the next block at as close to the same path velocity as possible.

Contour

Contour of the → workpiece

Contour monitoring

The following error is monitored within a definable tolerance band as a measure of contour accuracy. An unacceptably high following error can cause the drive to become overloaded, for example. In such cases, an alarm is output and the axes are stopped.

Coordinate system

See → Machine coordinate system, → Workpiece coordinate system

CPU

Central processing unit, see → PLC

Curvature

The curvature k of a contour is the inverse of radius r of the nestling circle in a contour point (k = 1/r).

Cycles

Protected subprograms for execution of repetitive machining operations on the → workpiece.

Data block

1. Data unit of the → PLC that → HIGHSTEP programs can access.
2. Data unit of the → NC: Data modules contain data definitions for global user data. This data can be initialized directly when it is defined.

Data word

Two-byte data unit within a → data block.

Diagnostics

1. Operating area of the controller.
2. The controller has a self-diagnostics program as well as test functions for servicing purposes: status, alarm, and service displays

Dimensions specification, metric and inches

Position and pitch values can be programmed in inches in the machining program. Irrespective of the programmable dimensions (G70/G71), the controller is set to a basic system.

DRF

Differential Resolver Function: NC function which generates an incremental zero offset in Automatic mode in conjunction with an electronic handwheel.

Drive

The drive is the unit of the CNC that performs the speed and torque control based on the settings of the NC.

Dynamic feedforward control

Inaccuracies in the → contour due to following errors can be practically eliminated using dynamic, acceleration-dependent feedforward control. This results in excellent machining accuracy even at high → path velocities. Feedforward control can be selected and deselected on an axis-specific basis via the → part program.

Editor

The editor makes it possible to create, edit, extend, join, and import programs/texts/program blocks.

Exact stop

When an exact stop statement is programmed, the position specified in a block is approached exactly and, if necessary, very slowly. To reduce the approach time, → exact stop limits are defined for rapid traverse and feed.

Exact stop limit

When all path axes reach their exact stop limits, the controller responds as if it had reached its precise destination point. A block advance of the → part program occurs.

External zero offset

Zero offset specified by the → PLC.

Fast retraction from the contour

When an interrupt occurs, a motion can be initiated via the CNC machining program, enabling the tool to be quickly retracted from the workpiece contour that is currently being machined. The retraction angle and the distance retracted can also be parameterized. An interrupt routine can also be executed following the fast retraction.

Feed override

The programmed velocity is overriden by the current velocity setting made via the → machine control panel or from the → PLC (0 to 200%). The feedrate can also be corrected by a programmable percentage factor (1 to 200%) in the machining program.

Finished-part contour

Contour of the finished workpiece. See → Raw part.

Fixed machine point

Point that is uniquely defined by the machine tool, e.g. machine reference point.

Fixed-point approach

Machine tools can approach fixed points such as a tool change point, loading point, pallet change point, etc. in a defined way. The coordinates of these points are stored in the controller. The controller moves the relevant axes in → rapid traverse, whenever possible.

Frame

A frame is an arithmetic rule that transforms one Cartesian coordinate system into another Cartesian coordinate system. A frame contains the following components: → zero offset, → rotation, → scaling, → mirroring.

Functionality

The path-jerk limitation can be activated/deactivated by programming the setting data.

Parameter: *Value*

- Value range: TRUE, FALSE

Application:

- Part program
- Static synchronized action

Geometry

Description of a → workpiece in the → workpiece coordinate system.

Geometry axis

Geometry axes are used to describe a 2- or 3-dimensional area in the workpiece coordinate system.

Ground

Ground is taken as the total of all linked inactive parts of a device which will not become live with a dangerous contact voltage even in the event of a malfunction.

Helical interpolation

The helical interpolation function is ideal for machining internal and external threads using form milling cutters and for milling lubrication grooves.

The helix comprises two motions:

- Circular motion in one plane
- A linear motion perpendicular to this plane

High-level CNC language

The high-level language offers: → user-defined variables, → system variables, → macro techniques.

High-speed digital inputs/outputs

The digital inputs can be used for example to start fast CNC program routines (interrupt routines). High-speed, program-driven switching functions can be initiated via the digital CNC outputs

HIGHSTEP

Summary of programming options for → PLCs of the AS300/AS400 system.

HW Config

SIMATIC S7 tool for the configuration and parameterization of hardware components within an S7 project

Identifier

In accordance with DIN 66025, words are supplemented using identifiers (names) for variables (arithmetic variables, system variables, user variables), subprograms, key words, and words with multiple address letters. These supplements have the same meaning as the words with respect to block format. Identifiers must be unique. It is not permissible to use the same identifier for different objects.

Inch measuring system

Measuring system which defines distances in inches and fractions of inches.

Inclined surface machining

Drilling and milling operations on workpiece surfaces that do not lie in the coordinate planes of the machine can be performed easily using the function "inclined-surface machining".

Increment

Travel path length specification based on number of increments. The number of increments can be stored as → setting data or be selected by means of a suitably labeled key (i.e. 10, 100, 1000, 10000).

Incremental dimension

Also incremental dimension: A destination for axis traversal is defined by a distance to be covered and a direction referenced to a point already reached. See → Absolute dimension.

Intermediate blocks

Motions with selected → tool offset (G41/G42) may be interrupted by a limited number of intermediate blocks (blocks without axis motions in the offset plane), whereby the tool offset can still be correctly compensated for. The permissible number of intermediate blocks which the controller reads ahead can be set in system parameters.

Interpolator

Logic unit of the → NCK that defines intermediate values for the motions to be carried out in individual axes based on information on the end positions specified in the part program.

Interpolatory compensation

Interpolatory compensation is a tool that enables manufacturing-related leadscrew error and measuring system error compensations (SSFK, MSFK).

Interrupt routine

Interrupt routines are special → subprograms that can be started by events (external signals) in the machining process. A part program block which is currently being worked through is interrupted and the position of the axes at the point of interruption is automatically saved.

Inverse-time feedrate

The time required for the path of a block to be traversed can also be programmed for the axis motion instead of the feed velocity (G93).

JOG

Control operating mode (setup mode): In JOG mode, the machine can be set up. Individual axes and spindles can be traversed in JOG mode by means of the direction keys. Additional functions in JOG mode include: → Reference point approach, → Repos, and → Preset (set actual value).

Key switch

The key switch on the → machine control panel has four positions that are assigned functions by the operating system of the controller. The key switch has three different colored keys that can be removed in the specified positions.

Keywords

Words with specified notation that have a defined meaning in the programming language for → part programs.

KÜ

Transformation ratio

KV

Servo gain factor, a control variable in a control loop.

Leading axis

The leading axis is the → gantry axis that exists from the point of view of the operator and programmer and, thus, can be influenced like a standard NC axis.

Leadscrew error compensation

Compensation for the mechanical inaccuracies of a leadscrew participating in the feed. The controller uses stored deviation values for the compensation.

Limit speed

Maximum/minimum (spindle) speed: The maximum speed of a spindle can be limited by specifying machine data, the → PLC or → setting data.

Linear axis

In contrast to a rotary axis, a linear axis describes a straight line.

Linear interpolation

The tool travels along a straight line to the destination point while machining the workpiece.

Load memory

The load memory is the same as the → working memory for the CPU 314 of the → PLC.

Look Ahead

The **Look Ahead** function is used to achieve an optimal machining speed by looking ahead over an assignable number of traversing blocks.

Machine axes

Physically existent axes on the machine tool.

Machine control panel

An operator panel on a machine tool with operating elements such as keys, rotary switches, etc., and simple indicators such as LEDs. It is used to directly influence the machine tool via the PLC.

Machine coordinate system

A coordinate system, which is related to the axes of the machine tool.

Machine zero

Fixed point of the machine tool to which all (derived) measuring systems can be traced back.

Machining channel

A channel structure can be used to shorten idle times by means of parallel motion sequences, e.g. moving a loading gantry simultaneously with machining. Here, a CNC channel must be regarded as a separate CNC control system with decoding, block preparation and interpolation.

Macro techniques

Grouping of a set of statements under a single identifier. The identifier represents the set of consolidated statements in the program.

Main block

A block prefixed by ":" introductory block, containing all the parameters required to start execution of a -> part program.

Main program

The term "main program" has its origins during the time when part programs were split strictly into main and → subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program in the channel can be selected and started. It then runs through in → program level 0 (main program level). Further part programs or → cycles as subprograms can be called up in the main program.

MDA

Control operating mode: Manual Data Automatic. In the MDA mode, individual program blocks or block sequences with no reference to a main program or subprogram can be input and executed immediately afterwards through actuation of the NC start key.

Messages

All messages programmed in the part program and → alarms detected by the system are displayed on the operator panel in plain text with date and time and the corresponding symbol for the cancel criterion. Alarms and messages are displayed separately.

Metric measuring system

Standardized system of units: For length, e.g. mm (millimeters), m (meters).

Mirroring

Mirroring reverses the signs of the coordinate values of a contour, with respect to an axis. It is possible to mirror with respect to more than one axis at a time.

Mode

An operating concept on a SINUMERIK controller. The following modes are defined: → Jog, → MDA, → Automatic.

Mode group

Axes and spindles that are technologically related can be combined into one mode group. Axes/spindles of a mode group can be controlled by one or more → channels. The same → mode type is always assigned to the channels of the mode group.

NC

Numerical Control: Numerical control (NC) includes all components of machine tool control: → NCK, → PLC, HMI, → COM.

Note

A more correct term for SINUMERIK controllers would be: Computerized Numerical Control

NCK

Numerical Control Kernel: Component of NC that executes the → part programs and basically coordinates the motion operations for the machine tool.

Network

A network is the connection of multiple S7-300 and other end devices, e.g. a programming device via a → connecting cable. A data exchange takes place over the network between the connected devices.

NRK

Numeric robotic kernel (operating system of → NCK)

NURBS

The motion control and path interpolation that occurs within the controller is performed based on NURBS (**N**on **U**niform **R**ational **B**-**S**plines). This provides a uniform procedure for all internal interpolations.

OEM

The scope for implementing individual solutions (OEM applications) has been provided for machine manufacturers, who wish to create their own user interface or integrate technology-specific functions in the controller.

Offset memory

Data range in the control, in which the tool offset data is stored.

Oriented spindle stop

Stops the workpiece spindle in a specified angular position, e.g. in order to perform additional machining at a particular location.

Oriented tool retraction

RETTOOL: If machining is interrupted (e.g. when a tool breaks), a program command can be used to retract the tool in a user-specified orientation by a defined distance.

Overall reset

In the event of an overall reset, the following memories of the → CPU are deleted:

- → Working memory
- Read/write area of → load memory
- → System memory
- → Backup memory

Override

Manual or programmable control feature which enables the user to override programmed feedrates or speeds in order to adapt them to a specific workpiece or material.

Part program

Series of statements to the NC that act in concert to produce a particular → workpiece. Likewise, this term applies to execution of a particular machining operation on a given → raw part.

Part program block

Part of a → part program that is demarcated by a line feed. There are two types: → main blocks and → subblocks.

Part program management

Part program management can be organized by → workpieces. The size of the user memory determines the number of programs and the amount of data that can be managed. Each file (programs and data) can be given a name consisting of a maximum of 24 alphanumeric characters.

Path axis

Path axes include all machining axes of the → channel that are controlled by the → interpolator in such a way that they start, accelerate, stop, and reach their end point simultaneously.

Path feedrate

Path feed affects → path axes. It represents the geometric sum of the feedrates of the → geometry axes involved.

Path velocity

The maximum programmable path velocity depends on the input resolution. For example, with a resolution of 0.1 mm the maximum programmable path velocity is 1000 m/min.

PCIN data transfer program

PCIN is an auxiliary program for sending and receiving CNC user data (e.g. part programs, tool offsets, etc.) via a serial interface. The PCIN program can run in MS-DOS on standard industrial PCs.

Peripheral module

I/O modules represent the link between the CPU and the process.

I/O modules are:

- → Digital input/output modules
- → Analog input/output modules
- → Simulator modules

PLC

Programmable **L**ogic **C**ontroller: → Programmable logic controller. Component of → NC: Programmable control for processing the control logic of the machine tool.

PLC program memory

SINUMERIK 840D sl: The PLC user program, the user data and the basic PLC program are stored together in the PLC user memory.

PLC programming

The PLC is programmed using the **STEP 7** software. The STEP 7 programming software is based on the **WINDOWS** standard operating system and contains the STEP 5 programming functions with innovative enhancements.

Polar coordinates

A coordinate system which defines the position of a point on a plane in terms of its distance from the origin and the angle formed by the radius vector with a defined axis.

Polynomial interpolation

Polynomial interpolation enables a wide variety of curve characteristics to be generated, such as **straight line, parabolic, exponential functions** (SINUMERIK 840D sl).

Positioning axis

Axis that performs an auxiliary motion on a machine tool (e.g. tool magazine, pallet transport). Positioning axes are axes that do not interpolate with → path axes.

Pre-coincidence

Block change occurs already when the path distance approaches an amount equal to a specifiable delta of the end position.

Program block

Program blocks contain the main program and subprograms of → part programs.

Program level

A part program started in the channel runs as a → main program on program level 0 (main program level). Any part program called up in the main program runs as a → subprogram on a program level 1 ... n of its own.

Programmable frames

Programmable → frames enable dynamic definition of new coordinate system output points while the part program is being executed. A distinction is made between absolute definition using a new frame and additive definition with reference to an existing starting point.

Programmable logic controller

Programmable logic controllers (PLCs) are electronic controllers, the function of which is stored as a program in the control unit. This means that the layout and wiring of the device do not depend on the function of the controller. The programmable logic control has the same structure as a computer; it consists of a CPU (central module) with memory, input/output modules and an internal bus system. The peripherals and the programming language are matched to the requirements of the control technology.

Programmable working area limitation

Limitation of the motion space of the tool to a space defined by programmed limitations.

Programming key

Characters and character strings that have a defined meaning in the programming language for → part programs.

Protection zone

Three-dimensional zone within the → working area into which the tool tip must not pass.

Quadrant error compensation

Contour errors at quadrant transitions, which arise as a result of changing friction conditions on the guideways, can be virtually entirely eliminated with the quadrant error compensation. Parameterization of the quadrant error compensation is performed by means of a circuit test.

R parameters

Arithmetic parameter that can be set or queried by the programmer of the → part program for any purpose in the program.

Rapid traverse

The highest traverse rate of an axis. For example, rapid traverse is used when the tool approaches the → workpiece contour from a resting position or when the tool is retracted from the workpiece contour. The rapid traverse velocity is set on a machine-specific basis using a machine data element.

Reference point

Machine tool position that the measuring system of the → machine axes references.

Rotary axis

Rotary axes apply a workpiece or tool rotation to a defined angular position.

Rotation

Component of a → frame that defines a rotation of the coordinate system around a particular angle.

Rounding axis

Rounding axes rotate a workpiece or tool to an angular position corresponding to an indexing grid. When a grid index is reached, the rounding axis is "in position".

RS-232-C

Serial interface for data input/output. Machining programs as well as manufacturer and user data can be loaded and saved via this interface.

Safety functions

The controller is equipped with permanently active monitoring functions that detect faults in the → CNC, the → PLC, and the machine in a timely manner so that damage to the workpiece, tool, or machine is largely prevented. In the event of a fault, the machining operation is interrupted and the drives stopped. The cause of the malfunction is logged and output as an alarm. At the same time, the PLC is notified that a CNC alarm has been triggered.

Scaling

Component of a → frame that implements axis-specific scale modifications.

Setting data

Data which communicates the properties of the machine tool to the NC as defined by the system software.

Softkey

A key, whose name appears on an area of the screen. The choice of softkeys displayed is dynamically adapted to the operating situation. The freely assignable function keys (softkeys) are assigned defined functions in the software.

Software limit switch

Software limit switches limit the traversing range of an axis and prevent an abrupt stop of the slide at the hardware limit switch. Two value pairs can be specified for each axis and activated separately by means of the → PLC.

Spline interpolation

With spline interpolation, the controller can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

Standard cycles

Standard cycles are provided for machining operations which are frequently repeated:

- For the drilling/milling technology
- For turning technology

The available cycles are listed in the "Cycle support" menu in the "Program" operating area. Once the desired machining cycle has been selected, the parameters required for assigning values are displayed in plain text.

Subblock

Block preceded by "N" containing information for a sequence, e.g. positional data.

Subprogram

The term "subprogram" has its origins during the time when part programs were split strictly into →main and subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program or any → cycle can be called up as a subprogram within another part program. It then runs through in the next → program level (x+1) (subprogram level (x+1)).

Synchronization

Statements in → part programs for coordination of sequences in different → channels at certain machining points.

Synchronized actions

1. Auxiliary function output

   During workpiece machining, technological functions (→ auxiliary functions) can be output from the CNC program to the PLC. For example, these auxiliary functions are used to control additional equipment for the machine tool, such as quills, grabbers, clamping chucks, etc.

2. Fast auxiliary function output

   For time-critical switching functions, the acknowledgement times for the → auxiliary functions can be minimized and unnecessary hold points in the machining process can be avoided.

Synchronized axes

Synchronized axes take the same time to traverse their path as the geometry axes take for their path.

Synchronized axis

A synchronized axis is the → gantry axis whose set position is continuously derived from the motion of the → leading axis and is, thus, moved synchronously with the leading axis. From the point of view of the programmer and operator, the synchronized axis "does not exist".

Syntax

$SC_IS_SD_MAX_PATH_JERK = *value*

System memory

The system memory is a memory in the CPU in which the following data is stored:

- Data required by the operating system
- The operands timers, counters, markers

System variable

A variable that exists without any input from the programmer of a → part program. It is defined by a data type and the variable name preceded by the character $. See → User-defined variable.

Tapping without compensating chuck

This function allows threads to be tapped without a compensating chuck. By using the interpolating method of the spindle as a rotary axis and the drilling axis, threads can be cut to a precise final drilling depth, e.g. for blind hole threads (requirement: spindles in axis operation).

Text editor

See → Editor

TOA area

The TOA area includes all tool and magazine data. By default, this area coincides with the → channel area with regard to the access of the data. However, machine data can be used to specify that multiple channels share one → TOA unit so that common tool management data is then available to these channels.

TOA unit

Each → TOA area can have more than one TOA unit. The number of possible TOA units is limited by the maximum number of active → channels. A TOA unit includes exactly one tool data block and one magazine data block. In addition, a TOA unit can also contain a toolholder data block (optional).

Tool

Active part on the machine tool that implements machining (e.g. turning tool, milling tool, drill, LASER beam, etc.).

Tool nose radius compensation

Contour programming assumes that the tool is pointed. Because this is not actually the case in practice, the curvature radius of the tool used must be communicated to the controller which then takes it into account. The curvature center is maintained equidistantly around the contour, offset by the curvature radius.

Tool offset

Consideration of the tool dimensions in calculating the path.

Tool radius compensation

To directly program a desired → workpiece contour, the control must traverse an equistant path to the programmed contour taking into account the radius of the tool that is being used (G41/G42).

Transformation

Additive or absolute zero offset of an axis.

Travel range

The maximum permissible travel range for linear axes is ± 9 decades. The absolute value depends on the selected input and position control resolution and the unit of measurement (inch or metric).

User interface

The user interface (UI) is the display medium for a CNC in the form of a screen. It features horizontal and vertical softkeys.

User memory

All programs and data, such as part programs, subprograms, comments, tool offsets, and zero offsets/frames, as well as channel and program user data, can be stored in the shared CNC user memory.

User program

User programs for the S7-300 automation systems are created using the programming language STEP 7. The user program has a modular layout and consists of individual blocks.

The basic block types are:

- Code blocks

  These blocks contain the STEP 7 commands.

- Data blocks

  These blocks contain constants and variables for the STEP 7 program.

User-defined variable

Users can declare their own variables for any purpose in the → part program or data block (global user data). A definition contains a data type specification and the variable name. See → System variable.

Variable definition

A variable definition includes the specification of a data type and a variable name. The variable names can be used to access the value of the variables.

Velocity control

In order to achieve an acceptable traverse rate in the case of very slight motions per block, an anticipatory evaluation over several blocks (→ Look Ahead) can be specified.

WinSCP

WinSCP is a freely available open source program for Windows for the transfer of files.

Working area

Three-dimensional zone into which the tool tip can be moved on account of the physical design of the machine tool. See → Protection zone.

Working area limitation

With the aid of the working area limitation, the traversing range of the axes can be further restricted in addition to the limit switches. One value pair per axis may be used to describe the protected working area.

Working memory

The working memory is a RAM in the → CPU that the processor accesses when processing the application program.

Workpiece

Part to be made/machined by the machine tool.

Workpiece contour

Set contour of the → workpiece to be created or machined.

Workpiece coordinate system

The workpiece coordinate system has its starting point in the → workpiece zero-point. In machining operations programmed in the workpiece coordinate system, the dimensions and directions refer to this system.

Workpiece zero

The workpiece zero is the starting point for the → workpiece coordinate system. It is defined in terms of distances to the → machine zero.

Zero offset

Specifies a new reference point for a coordinate system through reference to an existing zero point and a → frame.

1. Settable

   A configurable number of settable zero offsets are available for each CNC axis. The offsets - which are selected by means of G functions - take effect alternatively.

2. External

   In addition to all the offsets which define the position of the workpiece zero, an external zero offset can be overridden by means of the handwheel (DRF offset) or from the PLC.

3. Programmable

   Zero offsets can be programmed for all path and positioning axes using the `TRANS` statement.

Index

3

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z